NSIS Users Manual

Check http://nsis.sf.net for news, information, support, examples, tutorials and more.

Quick links:

FAQ - A list of frequently asked questions NSIS Wiki - Examples, functions, tutorials, plug-ins, software and more Forum - Post questions or discuss NSIS features

Copyright (C) 1999-2016 Contributors

- Chapter 1: Introduction to NSIS
 - About NSIS
 - Main Features
 - Feature List
 - Unicode installers
- Chapter 2: Tutorial: The Basics
 - Introduction
 - Script Files
 - Scripting structure
 - Installer Attributes
 - Pages
 - Sections
 - Functions
 - Working with Scripts
 - Logical Code Structures
 - Variables
 - Debugging Scripts
 - Script Execution
 - Compiler Commands
 - Compiler
 - Modern UI
 - Plug-ins
 - More
- Chapter 3: Command Line Usage
 - MakeNSIS Usage

- Options
- Notes
- Environment variables
- Examples
- Installer Usage
 - Common Options
 - Uninstaller Specific Options
 - Examples

• Chapter 4: Scripting Reference

- Script File Format
- Variables
 - User Variables
 - Var
 - Other Writable Variables
 - Constants
 - Constants Used in Strings
- Labels
- Relative Jumps
- Pages
 - Ordering
 - Page Options
 - Callbacks
 - Page
 - UninstPage
 - PageEx
 - PageExEnd
 - PageCallbacks
- Sections
 - Section Commands
 - AddSize
 - Section
 - SectionEnd
 - SectionIn
 - SectionGroup
 - SectionGroupEnd
 - Uninstall Section
- Functions

- Function Commands
 - Function
 - FunctionEnd
- Callback Functions
 - Install Callbacks
 - .onGUIInit
 - .onInit
 - .onInstFailed
 - .onInstSuccess
 - .onGUIEnd
 - .onMouseOverSection
 - .onRebootFailed
 - .onSelChange
 - .onUserAbort
 - .onVerifyInstDir
 - Uninstall Callbacks
 - un.onGUIInit
 - un.onInit
 - un.onUninstFailed
 - un.onUninstSuccess
 - un.onGUIEnd
 - un.onRebootFailed
 - un.onSelChange
 - un.onUserAbort
- Installer Attributes
 - General Attributes
 - AddBrandingImage
 - AllowRootDirInstall
 - AutoCloseWindow
 - BGFont
 - BGGradient
 - BrandingText
 - Caption
 - ChangeUI
 - CheckBitmap
 - CompletedText
 - ComponentText

- CRCCheck
- DetailsButtonText
- DirText
- DirVar
- DirVerify
- FileErrorText
- Icon
- InstallButtonText
- InstallColors
- InstallDir
- InstallDirRegKey
- InstProgressFlags
- InstType
- LicenseBkColor
- LicenseData
- LicenseForceSelection
- LicenseText
- ManifestDPIAware
- ManifestSupportedOS
- MiscButtonText
- Name
- OutFile
- RequestExecutionLevel
- SetFont
- ShowInstDetails
- ShowUninstDetails
- SilentInstall
- SilentUnInstall
- SpaceTexts
- SubCaption
- UninstallButtonText
- UninstallCaption
- UninstallIcon
- UninstallSubCaption
- UninstallText
- Windowlcon
- XPStyle

- Compiler Flags
 - AllowSkipFiles
 - FileBufSize
 - SetCompress
 - SetCompressor
 - SetCompressorDictSize
 - SetDatablockOptimize
 - SetDateSave
 - SetOverwrite
 - Unicode
- Version Information
 - VIAddVersionKey
 - VIProductVersion
 - VIFileVersion
- Instructions
 - Basic Instructions
 - Delete
 - Exec
 - ExecShell
 - ExecWait
 - File
 - Rename
 - ReserveFile
 - RMDir
 - SetOutPath
 - Registry, INI, File Instructions
 - DeleteINISec
 - DeleteINIStr
 - DeleteRegKey
 - DeleteRegValue
 - EnumRegKey
 - EnumRegValue
 - ExpandEnvStrings
 - FlushINI
 - ReadEnvStr
 - ReadINIStr
 - ReadRegDWORD

- ReadRegStr
- WriteINIStr
- WriteRegBin
- WriteRegDWORD
- WriteRegStr
- WriteRegExpandStr
- General Purpose Instructions
 - CallInstDLL
 - CopyFiles
 - CreateDirectory
 - CreateShortcut
 - GetDLLVersion
 - GetDLLVersionLocal
 - GetFileTime
 - GetFileTimeLocal
 - GetFullPathName
 - GetTempFileName
 - SearchPath
 - SetFileAttributes
 - RegDLL
 - UnRegDLL
- Flow Control Instructions
 - Abort
 - Call
 - ClearErrors
 - GetCurrentAddress
 - GetFunctionAddress
 - GetLabelAddress
 - Goto
 - IfAbort
 - IfErrors
 - IfFileExists
 - IfRebootFlag
 - IfSilent
 - IntCmp
 - IntCmpU
 - MessageBox

- Return
- Quit
- SetErrors
- StrCmp
- StrCmpS
- File Instructions
 - FileClose
 - FileOpen
 - FileRead
 - FileReadUTF16LE
 - FileReadByte
 - FileReadWord
 - FileSeek
 - FileWrite
 - FileWriteUTF16LE
 - FileWriteByte
 - FileWriteWord
 - FindClose
 - FindFirst
 - FindNext
- Uninstaller Instructions
 - WriteUninstaller
- Miscellaneous Instructions
 - GetErrorLevel
 - GetInstDirError
 - InitPluginsDir
 - Nop
 - SetErrorLevel
 - SetRegView
 - SetShellVarContext
 - Sleep
- String Manipulation Instructions
 - StrCpy
 - StrLen
- Stack Support
 - Exch
 - Pop

- Push
- Integer Support
 - IntFmt
 - IntOp
- Reboot Instructions
 - Reboot
 - SetRebootFlag
- Install Logging Instructions
 - LogSet
 - LogText
- Section Management
 - SectionSetFlags
 - SectionGetFlags
 - SectionSetText
 - SectionGetText
 - SectionSetInstTypes
 - SectionGetInstTypes
 - SectionSetSize
 - SectionGetSize
 - SetCurInstType
 - GetCurInstType
 - InstTypeSetText
 - InstTypeGetText
- User Interface Instructions
 - BringToFront
 - CreateFont
 - DetailPrint
 - EnableWindow
 - FindWindow
 - GetDlgItem
 - HideWindow
 - IsWindow
 - LockWindow
 - SendMessage
 - SetAutoClose
 - SetBrandingImage
 - SetDetailsView

- SetDetailsPrint
- SetCtlColors
- SetSilent
- ShowWindow
- Multiple Languages Instructions
 - LoadLanguageFile
 - LangString
 - LicenseLangString
- Multiple Languages
 - Language Selection
 - LangDLL Plug-in
 - RTL Languages
- Plug-in DLLs
 - Using Plug-in Commands
 - Calling plug-ins manually
- Silent Installers/Uninstallers
- Chapter 5: Compile Time Commands
 - Compiler Utility Commands
 - !include
 - !addincludedir
 - !addplugindir
 - !appendfile
 - !cd
 - !delfile
 - !echo
 - !error
 - !execute
 - !makensis
 - !packhdr
 - !finalize
 - !system
 - !tempfile
 - !getdllversion
 - !warning
 - !verbose
 - Predefines
 - \${__COUNTER__}

- \${ FILE }
- \${__FILEDIR__}
- \${ LINE }
- \${ DATE }
- \${__TIME__]
- \${__TIMESTAMP__}
- \${NSIS_VERSION}
- \${NSIS_PACKEDVERSION}
- \${NSIS CHAR SIZE}
- \${NSIS PTR SIZE}
- \${U+1}...\${U+10FFFF}
- Scope Predefines
 - \${ GLOBAL__}
 - \${ SECTION
 - \${ FUNCTION
 - \${ PAGEEX }
 - \${ UNINSTALL }
 - \${__MACRO__}
- Read environment variables
 - \$\%envVarName\%
- Conditional Compilation
 - !define
 - !undef
 - !ifdef
 - !ifndef
 - !if
 - !ifmacrodef
 - !ifmacrondef
 - !else
 - !endif
 - !insertmacro
 - !macro
 - !macroend
 - !macroundef
 - !searchparse
 - !searchreplace
- Appendix A: Modern User Interface

Appendix B: DLL/TLB Library Setup

- Introduction
- Library Installation
 - Introduction
 - Parameters
 - Options
 - LIBRARY X64
 - LIBRARY_SHELL_EXTENSION
 - LIBRARY COM
 - LIBRARY IGNORE VERSION
 - Notes
 - Example
 - Unshared DLL
 - Shared DLL
- Library Uninstallation
 - Introduction
 - Parameters
 - Options
 - LIBRARY X64
 - LIBRARY SHELL EXTENSION
 - LIBRARY_COM
 - Example
- Visual Basic 6 Run-Time Files

• Appendix C: Useful Scripts

- Get Internet Explorer version
- Is .NET Framework installed?
- Is Macromedia Flash Player installed?
- Connect to the Internet
- Get Installer Filename
- Prevent Multiple Instances
- More

• Appendix D: Useful Information

- Error Levels
- Add uninstall information to Add/Remove Programs
- Calling an external DLL using the System.dll plug-in
- Dump Content of Log Window to File
- How to Read REG_MULTI_SZ Values

- Predefined Macros for Unicode support
- Appendix E: Useful Headers
 - File Functions Header
 - Introduction
 - Locate
 - GetSize
 - DriveSpace
 - GetDrives
 - GetTime
 - GetFileAttributes
 - GetFileVersion
 - GetExeName
 - GetExePath
 - GetParameters
 - GetOptions
 - GetOptionsS
 - GetRoot
 - GetParent
 - GetFileName
 - GetBaseName
 - GetFileExt
 - BannerTrimPath
 - DirState
 - RefreshShellIcons
 - Text Functions Header
 - Introduction
 - LineFind
 - LineRead
 - FileReadFromEnd
 - LineSum
 - FileJoin
 - TextCompare
 - TextCompareS
 - ConfigRead
 - ConfigReadS
 - ConfigWrite
 - ConfigWriteS

- FileRecode
- TrimNewLines
- Word Functions Header
 - Introduction
 - WordFind
 - WordFindS
 - WordFind2X
 - WordFind2XS
 - WordFind3X
 - WordFind3XS
 - WordReplace
 - WordReplaceS
 - WordAdd
 - WordAddS
 - WordInsert
 - WordInsertS
 - StrFilter
 - StrFilterS
 - VersionCompare
 - VersionConvert
- Appendix F: Changelog and Release Notes
 - 0 3.0
 - Release Notes
 - Changelog
 - Minor Changes
 - Translations
 - 3.0 Release Candidate 2
 - Changelog
 - Major Changes
 - Minor Changes
 - Translations
 - 3.0 Release Candidate 1
 - Changelog
 - Major Changes
 - Minor Changes
 - Translations
 - o 3.0 Beta 3

- Changelog
 - Major Changes
 - Minor Changes
 - Translations
- o 3.0 Beta 2
 - Changelog
 - Major Changes
 - Minor Changes
 - Build System
- o 3.0 Beta 1
 - Release Notes
 - Changelog
 - Major Changes
 - Minor Changes
 - Translations
- o 3.0 Beta 0
 - Release Notes
 - Changelog
 - Major Changes
 - Minor Changes
- o 3.0 Alpha 2
 - Release Notes
 - Known Issues
 - Changelog
 - Major Changes
 - Minor Changes
- o 3.0 Alpha 1
 - Release Notes
 - Known Issues
 - Changelog
 - Major Changes
 - Minor Changes
- o 3.0 Alpha 0
 - Release Notes
 - Known Issues
 - Changelog
 - Major Changes

- Minor Changes
- Translations
- o 2.51
 - Changelog
 - Minor Changes
 - Translations
- o 2.50
 - Changelog
 - Major Changes
 - Minor Changes
- o 2.49
 - Changelog
- o 2.48
 - Changelog
- o 2.47
 - Release Notes
 - Changelog
 - Major Changes
 - Minor Changes
- 0 2.46
 - Changelog
 - Minor Changes
 - Translations
 - Build System
- o 2.45
 - Release Notes
 - Changelog
 - Major Changes
 - Minor Changes
 - Modern UI
 - Translations
 - Build System
- 0 2.44
 - Changelog
 - Major Changes
 - Minor Changes
- o 2.43

- Release Notes
- Changelog
 - Minor Changes
 - Utilities and Plug-ins
 - Translations
 - Plug-in API
 - Build System
- 0 2.42
 - Release Notes
 - Changelog
 - Major Changes
 - Minor Changes
 - Translations
 - Build System
- 0 2.41
 - Changelog
 - Minor Changes
 - Translations
 - Build System
- 0 2.40
 - Changelog
 - Major Changes
 - Minor Changes
 - Translations
- 0 2.39
 - Changelog
 - Major Changes
 - Minor Changes
- 2.38
 - Changelog
 - Major Changes
 - Minor Changes
 - Modern UI
 - Translations
- o 2.37
 - Changelog
 - Major Changes

- Minor Changes
- Utilities and Plug-ins
- Translation
- Build System
- 2.36
 - Release Notes
 - Changelog
 - Major Changes
 - Modern UI
 - nsDialogs
 - Minor Changes
 - Utilities and Plug-ins
 - Translation
 - Build System
- o 2.35
 - Changelog
 - Major Changes
 - Minor Changes
 - Utilities and Plug-ins
- 0 2.34
 - Release Notes
 - Changelog
 - Major Changes
 - Minor Changes
 - Utilities and Plug-ins
 - Translations
 - Build System
- 0 2.33
 - Release Notes
 - Changelog
 - Major Changes
 - Minor Changes
 - Utilities and Plug-ins
 - Translations
 - Build System
- o 2.32
 - Release Notes

- Changelog
 - Major Changes
 - Minor Changes
 - Utilities and Plug-ins
 - Translations
 - Build System
- o 2.31
 - Release Notes
 - Changelog
 - Minor Changes
 - Utilities and Plug-ins
 - Translations
 - Build System
- 0 2.30
 - Release Notes
 - Changelog
 - Major Changes
 - Minor Changes
 - Utilities and Plug-ins
 - Translations
 - Build System
- 0 2.29
 - Release Notes
 - Changelog
 - Major Changes
 - Minor Changes
 - Utilities and Plug-ins
 - Translations
 - Build System
- o 2.28
 - Changelog
 - Major Changes
 - Minor Changes
 - Utilities and Plug-ins
 - Translations
 - Build System
- o 2.27

- Release Notes
- Changelog
 - Major Changes
 - Minor Changes
 - Build System
- 0 2.26
 - Release Notes
 - Changelog
 - Major Changes
 - Minor Changes
 - New/Changed Commands
 - Utilities and Plug-ins
 - Translations
 - Build System
- 0 2.25
 - Changelog
 - Major Changes
 - Minor Changes
 - New/Changed Commands
 - Utilities and Plug-ins
 - Translations
 - Build System
- o 2.24
 - Changelog
 - Major Changes
 - Minor Changes
 - New/Changed Commands
 - Utilities and Plug-ins
 - Translations
 - Build System
- 0 2.23
 - Changelog
 - Minor Changes
 - Utilities and Plug-ins
 - Build System
- 0 2.22
 - Release Notes

- Changelog
 - Minor Changes
 - New/Changed Commands
 - Translations
 - Build System
- 0 2.21
 - Changelog
 - Major Changes
 - Minor Changes
 - New/Changed Commands
 - Translations
 - Build System
- 0 2.20
 - Changelog
 - Minor Changes
 - Translations
 - Build System
- 0 2.19
 - Changelog
 - Minor Changes
 - New/Changed Commands
 - Plug-ins
 - Translations
- o 2.18
 - Changelog
 - Minor Changes
 - Translations
- o 2.17
 - Changelog
 - Minor Changes
 - Translations
- 2.16
 - Release Notes
 - Changelog
 - Major Changes
 - New/Changed Commands
 - Minor Changes

- Translations
- Build System
- 2.15
 - Changelog
 - New/Changed Commands
 - Minor Changes
 - Translations
 - Build System
- 0 2.14
 - Release Notes
 - Changelog
 - Major Changes
 - Minor Changes
- 2.13
 - Release Notes
 - Changelog
 - Major Changes
 - New/Changed Commands
 - Minor Changes
 - Utilities and Plug-ins
 - Translations
 - Build System
- o 2.12
 - Changelog
 - Major Changes
 - New/Changed Commands
 - Minor Changes
 - Utilities and Plug-ins
 - Translations
 - Build System
- o 2.11
 - Release Notes
 - Changelog
 - Major Changes
 - New/Changed Commands
 - Minor Changes
 - Utilities and Plug-ins

- Translations
- Build System
- 2.10
 - Changelog
 - Major Changes
 - Minor Changes
 - Translations
 - Build System
- o 2.09
 - Changelog
 - Major Changes
 - New/Changed Commands
 - Minor Changes
 - Translations
 - Build System
- o 2.08
 - Release Notes
 - Changelog
 - Major Changes
 - Minor Changes
 - Build System
- o 2.07
 - Release Notes
 - Changelog
 - Major Changes
 - New/Changed Commands
 - Minor Changes
 - Include Files
 - Utilities and Plug-ins
 - Translations
- o 2.06
 - Changelog
 - Major Changes
 - New/Changed Commands
 - Minor Changes
 - Utilities and Plug-ins
- o 2.05

- Release Notes
- Changelog
 - New/Changed Commands
 - Minor Changes
- 0 2.04
 - Changelog
 - Major Changes
 - Minor Changes
- o 2.03
 - Release Notes
 - Changelog
 - Major Changes
 - New/Changed Commands
 - Minor Changes
- 0 2.02
 - Changelog
 - Major Changes
 - New/Changed Commands
 - Minor Changes
 - Utilities and Plug-ins
- o 2.01
 - Release Notes
 - Changelog
 - Major Changes
 - New/Changed Commands
 - Minor Changes
 - Utilities and Plug-ins
- o 2.0
 - Release Notes
 - Changelog
 - Changes from 1.98
 - Changes from RC4
- 2.0 Release Candidate 4
 - Changelog
 - Major Changes
 - Minor Changes
 - Utilities and Plug-ins

- 2.0 Release Candidate 3
 - Changelog
 - Minor Changes
 - Utilities and Plug-ins
- 2.0 Release Candidate 2
 - Changelog
 - Minor Changes
 - Utilities and Plug-ins
- 2.0 Release Candidate 1
 - Release Notes
 - Changelog
 - Major Changes
 - Utilities and Plug-ins
- o 2.0 Beta 4
 - Release Notes
 - Changelog
 - Major Changes
 - New/Changed Commands
 - Minor Changes
 - Utilities and Plug-ins
- o 2.0 Beta 3
- o 2.0 Beta 2
- 2.0 Beta 1
- o 2.0 Beta 0
- o 2.0 Alpha 7
- 2.0 Alpha 6
- 2.0 Alpha 5
- o 2.0 Alpha 4
- 2.0 Alpha 3
- o 2.0 Alpha 2
- 2.0 Alpha 1
- o 2.0 Alpha 0
- Older Versions
- Appendix G: Building NSIS
 - Building in General
 - Building on Windows
 - Building on POSIX

- Nightly Builds
- Appendix H: Credits
 - Programmers
 - Testers
 - Designers
 - Translators
 - Writers
- Appendix I: License
 - Copyright
 - Applicable licenses
 - zlib/libpng license
 - bzip2 license
 - Common Public License version 1.0
 - Special exception for LZMA compression module

Chapter 1: Introduction to NSIS

- About NSIS
- Main Features
- Feature List
- Unicode installers

Chapter 2: Tutorial: The Basics

- Introduction
- Script Files
- Scripting structure
 - Installer Attributes
 - Pages
 - Sections
 - Functions
 - Working with Scripts
 - Script Execution
 - Compiler Commands
- Compiler
- Modern UI
- Plug-ins
- More

Chapter 3: Command Line Usage

- MakeNSIS Usage
 - Options
 - Notes
 - Environment variables
 - Examples
- Installer Usage
 - Common Options
 - Uninstaller Specific Options
 - Examples

Chapter 4: Scripting Reference

- Script File Format
- Variables
 - User Variables
 - Other Writable Variables
 - Constants
 - Constants Used in Strings
- Labels
- Relative Jumps
- Pages
 - Ordering
 - Page Options
 - Callbacks
 - Page
 - UninstPage
 - PageEx
 - PageExEnd
 - PageCallbacks
- Sections
 - Section Commands
 - Uninstall Section
- Functions
 - Function Commands
 - Callback Functions
- Installer Attributes
 - General Attributes
 - Compiler Flags
 - Version Information
- Instructions
 - Basic Instructions
 - Registry, INI, File Instructions
 - General Purpose Instructions
 - Flow Control Instructions
 - File Instructions
 - Uninstaller Instructions

- Miscellaneous Instructions
- String Manipulation Instructions
- Stack Support
- Integer Support
- Reboot Instructions
- Install Logging Instructions
- Section Management
- User Interface Instructions
- Multiple Languages Instructions
- Multiple Languages
 - Language Selection
 - LangDLL Plug-in
 - RTL Languages
- Plug-in DLLs
 - Using Plug-in Commands
 - Calling plug-ins manually
- Silent Installers/Uninstallers

Chapter 5: Compile Time Commands

- Compiler Utility Commands
 - !include
 - !addincludedir
 - !addplugindir
 - !appendfile
 - o !cd
 - !delfile
 - !echo
 - !error
 - !execute
 - !makensis
 - !packhdr
 - !finalize
 - !system
 - !tempfile
 - !getdllversion
 - !warning
 - !verbose
- Predefines
 - \${__COUNTER__}
 - \${ FILE }
 - \${__FILEDIR__}
 - \${__LINE__}
 - \${__DATE__
 - \${__TIME__}
 - \${__TIMESTAMP__}
 - \${NSIS VERSION}
 - \${NSIS_PACKEDVERSION}
 - \${NSIS CHAR SIZE}
 - \${NSIS_PTR_SIZE}
 - \${U+1}...\${U+10FFFF}
 - Scope Predefines
- Read environment variables
 - \$%envVarName%

- Conditional Compilation
 - !define
 - !undef
 - !ifdef
 - !ifndef
 - o !if
 - !ifmacrodef
 - !ifmacrondef
 - !else
 - !endif
 - !insertmacro
 - !macro
 - !macroend
 - !macroundef
 - !searchparse
 - !searchreplace

1.1 About NSIS

The installer is your application's first impression. Slow or unsuccessful software installations is one of the most irritating computer problems. A quick and user friendly installer is therefore an essential part of your software product. NSIS (Nullsoft Scriptable Install System) is a tool that allows programmers to create such installers for Windows. It is released under an open source license and is completely free for any use.

NSIS creates installers that are capable of installing, uninstalling, setting system settings, extracting files, etc. Because it's based on script files you can fully control every part of your installer. The scripting language supports variables, functions and string manipulation, just like a normal programming language - but designed for the creation of installers. Even with all these features, NSIS is still the smallest installer system available. With the default options, it has an overhead of only 34 KB.

With NSIS 3 you can also create Unicode installers, targeting all the languages supported by the OS without display issues.

1.2 Main Features

Small overhead size

NSIS has been designed to be small, fast and efficient. While other installers often add hundreds of kilobytes or several megabytes to your installer data, a fully featured NSIS installer has a overhead of only 34 KB.

Compatible with all major Windows versions

You can create a single installer that is compatible with Windows 95, 98, ME, NT4, 2000, XP, 2003, Vista, Server 2008, 7, Server 2008R2, 8, Server 2012, 8.1, Server 2012R2 and Windows 10.

Unique compression methods

You can choose between three different integrated compression methods (ZLib, BZip2, LZMA). LZMA compression gives better results than any other common compression method. You don't have to use large self-extracting archive modules or other applications, the compression support is included in the 34 KB overhead.

Script based

Unlike other systems that can only generate installers based on a list of files and registry keys, NSIS has a powerful scripting language. This script language is designed for installers and has commands that help you to perform many installation tasks. You can easily add custom logic and handle different upgrades, version checks and more. On the NSIS Wiki you can find a lot more.

Multiple languages in one installer

One installer can support multiple interface languages. More than 60 translations are already included and you can also create your own language files. RTL (right-to-left) languages such as Arabic and Hebrew

are fully supported. Creating a Unicode native installer is also possible for even more supported languages.

Many features and checks for the target system

The script language provides commands you can use on the target system. From simple features like folder creation and registry editing to text/binary file modification, modification of environment variables and system reboots. By using one of the provided plug-ins you can even call the Windows API directly.

Custom dialogs and interfaces

You can create custom wizard pages to get user input or integrate configuration options. NSIS includes a classic and modern wizard interface, but it is even possible to create your own custom interface.

Plug-in system

NSIS can be extended with plug-ins that can communicate with the installer. They can be written in C, C++, Delphi or another language and can be used to perform installation tasks or extend the installer interface. You can use the plug-in with a single line of script code. Plug-ins are also be compressed like other installation data and will only be included when you are using their features.

Support for web installation, file patching

The NSIS distribution includes a set of plug-ins that allow you to download files from the internet, make internet connections, patch existing files and more.

Project integration, different releases and automatic builds

The NSIS compiler features a powerful preprocessor. This allows you to easily integrate multiple projects into a single installer or automatically generate installer builds. You can also generate different releases such as lite and full versions.

Easy and human readable file formats

The NSIS script format and the format used for interface dialogs are easy, documented and humanly readable, so you can edit your files with your favorite editor. This also makes automatic script generation possible.

1.3 Feature List

- Generates self contained executable installers
- Support for ZLIB, BZIP2 and LZMA data compression (files can be compressed individually or together)
- Uninstall support (installer can generate an uninstaller)
- Customizable user interface (dialogs, fonts, backgrounds, icons, text, checkmarks, images etc.)
- Classic and Modern wizard interface
- Fully multilingual, support for multiple languages in one installer.
 More than 60 translations are available, but you can also create your own. Unicode support allowing even more languages.
- Page system: You can add standard wizard pages or custom pages
- User selection of installation components, tree for component selection
- Multiple install configurations (usually Minimal, Typical, Full), and custom configuration
- Installer self-verification using a CRC32 checksum
- Small overhead over compressed data size (34 KB with default options)
- Ability to display a license agreement in text or RTF format
- Ability to detect destination directory from the registry
- Easy to use plug-in system (lots of plug-ins for creation of custom dialogs, internet connections, HTTP downloading, file patching, Win32 API calls etc. are included)
- Installers can be as large as 2GB
- Optional silent mode for automated installations
- A preprocessor with support for defined symbols, macro's, conditional compilation, standard predefines
- A lovely coding experience with elements of PHP and assembly (includes user variables, a stack, real flow control, etc.)
- Installers have their own VMs that let you write code that can support:
- File extraction (with configurable overwrite parameters)
- File/directory copying, renaming, deletion, searching

- Plug-in DLL calling
- DLL/ActiveX control registration/deregistration
- Executable execution (shell execute and wait options)
- Shortcut creation
- Registry key reading/setting/enumerating/deleting
- INI file reading/writing
- Generic text file reading/writing
- Powerful string and integer manipulation
- Window finding based on class name or title
- User interface manipulation (font/text setting)
- Window message sending
- User interaction with message boxes or custom pages
- Branching, comparisons, etc.
- Error checking
- Reboot support, including delete or rename on reboot
- Installer behaviour commands (such as show/hide/wait/etc)
- User functions in script
- Callback functions for user actions
- Completely free for any use. See license.

1.4 Unicode installers

Starting with NSIS v3.0 you can choose to create Unicode installers by setting the Unicode attribute. These installers will not work on Windows 95/98/ME but they will allow you to display your installer in any Unicode language supported by the OS.

When building a Unicode installer NSIS variables can hold Unicode characters (0001-FFFF). There should be no need to modify your existing scripts. If you want to read/write Unicode files, specific instructions have been added to read/write UTF-16LE strings from/to disk.

2.1 Introduction

Most software packages you download or buy come with an installer. The installer copies and/or updates files, writes registry keys, writes configuration, creates shortcuts, etc. All of this is done automatically for the user. All the user needs to do is supply some information and the installer will do the rest. The user goes through a wizard, makes the appropriate choices and waits until the installer finishes. After the installer has finished the user is left only with the simple task of starting the program. The user doesn't have to worry about things he might have forgotten because all of the necessary steps were done by the installer.

NSIS is a tool for developers to create such installers. NSIS allows you to create everything from basic installers that just copies files to very complex installers that handle a lot of advanced tasks such as writing registry keys, settings environment variables, downloading the latest files from the internet, customizing configuration files and more. NSIS is very flexible and its scripting language is easy to learn.

NSIS compiles all of the files and the installation script into one executable file so your application will be easy to distribute. NSIS adds only about 34KB of code of its own (for the default configuration) to the data. NSIS boasts the smallest overhead available while still providing a lot of options thanks to its powerful scripting language and support of external plug-ins.

2.2 Script Files

To create a NSIS installer you first have to write a NSIS script. A NSIS script is just a regular text file with a special syntax. You can edit scripts with any text editor. It's recommended to use a text editor that shows line numbers because NSIS uses line numbers to indicate where errors lie, and to warn you about where errors might lie. An editor that supports syntax highlighting is also recommended. You can download editors made especially for NSIS and files for syntax highlighting from the NSIS Wiki.

In a NSIS script every line is treated as a command. If your command is too long for one line you can use a back-slash - '\' - at the end of the line. The compiler will treat the new line as an addition to the previous line and will not expect a new command. For example:

Messagebox MB_OK|MB_ICONINFORMATION \
"This is a sample that shows how to use line breaks fo

If you want to use a double-quote in a string you can either use \$\" to escape the quote or quote the string with a different type of quote such as ` or '.

For more details about the script format, see Script File Format.

The default extension for a script file is .nsi. Header files have the .nsh extension. Header files can help you arrange your script by dividing it to more than one block of code, you can also put functions or macros in header files and include the header files in multiple installers. This makes updating easier and it also makes your scripts easier to read. To include a header file in your script use !include. Header files that reside in the Include directory under your NSIS directory can be included just by their name. For example:

!include Sections.nsh

2.3 Scripting structure

A NSIS script contains Installer Attributes, Pages and Sections/Functions. You can also use Compiler Commands for compile-time operations. The OutFile instruction is required and tells NSIS where to write the installer, you also need at least one section.

2.3.1 Installer Attributes

Installer Attributes determine the behavior and the look and feel of your installer. With these attributes you can change texts that will be shown during the installation, the number of installation types etc. Most of these commands can only be set and are not changeable during runtime.

Other basic instructions are Name and InstallDir.

For more information about installer attributes, have a look at Installer Attributes.

2.3.2 Pages

A non-silent installer has a set of wizard pages to let the user configure the installer. You can set which pages to display using the Page command (or PageEx for more advanced settings). A typical set of pages looks like this:

```
Page license
Page components
Page directory
Page instfiles
UninstPage uninstConfirm
UninstPage instfiles
```

For the installer, this typical set of pages will display a license agreement, allow selection of components to install, allow selection of an installation directory, and finally install the selected components in the instfiles page.

For the uninstaller, it will display a confirmation page, and uninstall in the instfiles page.

2.3.3 Sections

It's common for installers to have several things the user can install. For example in the NSIS distribution installer you can choose to install additional tools, plug-ins, examples and more. Each of these components has its own piece of code. If the user selects to install this component then the installer will execute that code. In the script, that code is defined in sections. Each section corresponds to one component on the components page. The section's name is the displayed component name and the section code will be executed if that component is selected. It is possible to build your installer with only one section but if you want to use the components page and let the user choose what to install, you'll have to use more than one section.

Uninstallers can also have multiple sections. Uninstaller section names are prefixed with 'un.'. For example:

Section "Installer Section" SectionEnd

Section "un.Uninstaller Section" SectionFnd

The instructions that can be used in sections are very different from the installer attributes instructions, they are executed at runtime on the user's computer. Those instructions can extract files, read from and write to the registry, INI files or normal files, create directories, create shortcuts and a lot more. You can find out more in Instructions.

The most basic instructions are SetOutPath which tells the installer where to extract files and File which extracts files.

Example:

Section "My Program"

```
SetOutPath $INSTDIR
File "My Program.exe"
File "Readme.txt"
SectionEnd
```

For more information about sections see Sections.

2.3.4 Functions

Functions can contain script code, just like sections. The difference between sections and functions is the way they are called. There are two types of functions, user functions and callback functions.

User functions are called by the user from within sections or other functions using the Call instruction. User functions will not execute unless you call them. After the code in the function has executed the installer will continue executing the instructions that came after the Call instruction, unless you have aborted the installation inside the function. User functions are very useful if you have a set of instructions that need to be executed at several locations in the installers. If you put the code into a function you can save the copying time and you can maintain the code more easily.

Callback functions are called by the installer upon certain defined events such as when the installer starts. Callbacks are optional. If for example you want to welcome the user to your installer you can define a function called .onInit. The NSIS compiler will recognize this function as a callback function by the name and will call it when the installer starts.

```
Function .onInit
   MessageBox MB_YESNO "This will install My Program. D
   Abort
   gogogo:
FunctionEnd
```

Abort has a special meaning in callback functions. Each callback function has its own meaning for it, have a look at Callback Functions for more information. In the above example Abort tells the installer to stop

initializing the installer and quit immediately.

For more information about functions see Functions.

2.3.5 Working with Scripts

2.3.5.1 Logical Code Structures

Conditionally executing code, or executing code in a loop can be done using StrCmp, IntCmp, IfErrors, Goto and more. However, there's a much easier way do this. The LogicLib provides some very simple macros that allow easy construction of complex logical structures. Its syntax, explained in LogicLib.nsh, is similar to other programming languages and can prove to be simpler for beginners and advanced users alike.

For example, checking a value of a variable without the LogicLib can be done as follows.

```
StrCmp $0 'some value' 0 +3
   MessageBox MB_OK '$$0 is some value'
   Goto done
StrCmp $0 'some other value' 0 +3
   MessageBox MB_OK '$$0 is some other value'
   Goto done
# else
   MessageBox MB_OK '$$0 is "$0"'
done:
```

However, with the LogicLib the code is much more readable and easy to understand, as can be seen in the following example.

```
${If} $0 == 'some value'
  MessageBox MB_OK '$$0 is some value'
${ElseIf} $0 == 'some other value'
  MessageBox MB_OK '$$0 is some other value'
${Else}
  MessageBox MB_OK '$$0 is "$0"'
```

```
${EndIf}
```

The same can also be done using a switch, as shown in the following example.

```
${Switch} $0
  ${Case} 'some value'
    MessageBox MB_OK '$$0 is some value'
    ${Break}
  ${Case} 'some other value'
    MessageBox MB_OK '$$0 is some other value'
    ${Break}
  ${Default}
    MessageBox MB_OK '$$0 is "$0"'
    ${Break}
  ${EndSwitch}
```

Multiple conditions are also supported. The following example will notify the user, if both \$0 and \$1 are empty.

```
${If} $0 == ''
${AndIf} $1 == ''
MessageBox MB_OK|MB_ICONSTOP 'both are empty!'
${EndIf}
```

The LogicLib removes the need for labels and relative jumps, thus prevents label name conflicts, and removes the need to manually adjust relative jump offsets every time the script is changed.

It also simplifies looping by supporting the common while, do and for loops. All of the following examples count to five using the LogicLib.

```
StrCpy $R1 0
${While} $R1 < 5
   IntOp $R1 $R1 + 1
   DetailPrint $R1
${EndWhile}</pre>
```

```
${For} $R1 1 5
   DetailPrint $R1
${Next}

StrCpy $R1 0
${Do}
   IntOp $R1 $R1 + 1
   DetailPrint $R1
${LoopUntil} $R1 >= 5
```

To use the LogicLib the following line needs to be added near the top of the script.

```
!include LogicLib.nsh
```

More examples can be found in LogicLib.nsi.

2.3.5.2 Variables

You can declare your own variables (\$VARNAME) with the Var command. Variables are global and can be used in any Section or Function.

Declaring and using a user variable:

```
Var BLA ;Declare the variable

Section bla

StrCpy $BLA "123" ;Now you can use the variable $BLA

SectionEnd
```

In addition there is a stack, which can also be used for temporary storage. To access the stack use the commands Push and Pop. Push adds a value to the stack, Pop removes one and sets the variable.

For shared code, there are 20 registers available (like \$0 and \$R0). These static variables don't have to be declared and you won't get any name conflicts. If you want to use these variables in shared code, store the original values on the stack and restore the original values afterwards.

After calling the function, the variables contain the same value as before. Note the order when using multiple variables (last-in first-out):

```
Function bla

Push $R0
Push $R1
   ...code...

Pop $R1
Pop $R0
FunctionEnd
```

2.3.5.3 Debugging Scripts

The more you work with NSIS the more complex the scripts will become. This will increase the potential of mistakes, especially when dealing with lots of variables. There are a few possibilities to help you debugging the code. To display the contents of variables you should use MessageBoxes or DetailPrint. To get a brief overview about all variables you should use the plug-in DumpState. By default all actions of the Installer are printed out in the Log Window. You can access the log if you right-click in the Log Window and select "Copy Details To Clipboard". There is also a way to write it directly to a file, see here.

2.3.6 Script Execution

When a user runs an installer or uninstaller, pages are displayed in the order they were defined in the script. When the instfiles page is reached,

sections, corresponding to the selected components, are executed in the order they were defined in the script. If the components page is not displayed, all sections are executed, assuming they were not unselected or somehow disabled by the script.

Beside code in sections, there's also code in callback functions. If defined, they might be executed before the sections code. For example, the .onInit callback function is executed before anything else in the script. There are also page callback functions which are executed at certain points of the page display process.

2.3.7 Compiler Commands

Compiler commands will be executed at compile time on your computer. They can be used for conditional compilation, to include header files, to execute applications, to change the working directory and more. The most common usage is defines. Defines are compile time constants. You can define your product's version number and use it in your script. For example:

```
!define VERSION "1.0.3"
Name "My Program ${VERSION}"
OutFile "My Program Installer - ${VERSION}.exe"
```

For more information about defines see Conditional Compilation.

Another common use is macros. Macros are used to insert code at compile time, depending on defines and using the values of the defines. The macro's commands are inserted at compile time. This allows you to write a general code only once and use it a lot of times but with a few changes. For example:

```
!macro MyFunc UN
Function ${UN}MyFunc
  Call ${UN}DoRegStuff
  ReadRegStr $0 HKLM Software\MyProgram key
  DetailPrint $0
FunctionEnd
```

```
!macroend
!insertmacro MyFunc ""
!insertmacro MyFunc "un."
```

This macro helps you avoid writing the same code for both the installer and the uninstaller. The two !insertmacros insert two functions, one for the installer called MyFunc and one for the uninstaller called un.MyFunc and both do exactly the same thing.

For more information see Compile Time Commands.

2.4 Compiler

The second thing you need to do in order to create your installer after you have created your script is to compile your script. MakeNSIS.exe is the NSIS compiler. It reads your script, parses it and creates an installer for you.

To compile you can right-click your .nsi file and select Compile NSIS Script. This will cause MakeNSISW, the NSIS Compiler Interface, to launch and call MakeNSIS to compile your script. MakeNSISW receives the output of MakeNSIS and presents it to you in a window where you can see it, copy it, test the installer and more. Using makensis.exe from the command prompt is also possible.

The compiler will check your script and give you warnings or an error. If an error occurs (i.e. 2 parameters required but only 1 given) the compiler will abort and a short error message including the line number will be displayed. For non-critical errors the compiler will give a warning (i.e. two DirText commands in one script). If your script has no errors the compiler will output an installer for you to distribute.

NSIS supports different compression methods, as explained here. ZLIB is the default compression method, which is fast and uses only a little bit of memory. LZMA is a good method for the creation of small installers for internet distribution. BZIP2 usually compresses better than ZLIB but not as good as LZMA, it is useful if you need lower memory usage or fast script compilation.

It it also possible to compile Windows installers on Linux, BSD or Mac OS X servers. See Building NSIS for details.

2.5 Modern UI

A popular user interface for NSIS is the Modern User Interface. It has an interface like the wizards of recent Windows versions. The Modern UI is not only a customized resource file, it has a lots of new interface elements. It features a white header to describe the current step, a description area on the component page, a welcome page, a finish page that allows the user to run the application or reboot the system and more.

For more information, see the Modern UI 2 Readme and the Modern UI Examples.

2.6 Plug-ins

NSIS support plug-ins that can be called from the script. Plug-ins are DLL files written in C, C++, Delphi or another programming language and therefore provide a more powerful code base to NSIS.

A plug-in call looks like this:

```
DLLName::FunctionName "parameter number 1" "parameter
```

Every plug-in's function has its own requirements when it comes to parameters, some will require none, some will accept as many parameters as you want to send. Examples:

```
nsExec::ExecToLog '"${NSISDIR}\makensis.exe" /CMDHELP'
Pop $0 ; Process exit code or "error"
InstallOptions::dialog "$PLUGINSDIR\test.ini"
Pop $0 ; success/back/cancel/error
NSISdl::download http://download.nullsoft.com/winamp/c
Pop $0 ; "success" or a error code
```

The plug-ins that NSIS knows of are listed at the top of the compiler output (verbose level 4). NSIS searches for plug-ins in the Plugins folder under your NSIS directory and lists all of their available functions. You can use !addplugindir to tell NSIS to search in other directories too.

The NSIS distribution already includes many plug-ins. InstallOptions is a popular plug-in that allows you to create custom pages, in combination with the NSIS Page commands (See Pages). The Startmenu plug-in provides a page that allows the user to choose a Start Menu folder. There are a lot of plug-ins for different purposes, take a look in the Docs folder for help files and examples. You can find additional plug-ins online: NSIS Wiki.

You can also create a plug-in yourself. C/C++ and Delphi header files are already available, see the example plugin for how to do this. Source code of included plug-ins can also be found in the source code package.

2.7 More

This tutorial has described the basic NSIS features, to learn more about everything NSIS can do, take some time to read the rest of this manual.

3.1 MakeNSIS Usage

NSIS installers are generated by using the 'MakeNSIS' program to compile a NSIS script (.NSI) into an installer executable. The NSIS development kit installer sets up your computer so that you can compile a .nsi file by simply right-clicking on it in Explorer and selecting 'compile'.

If you want to use MakeNSIS on the command line, the syntax of makensis is:

makensis [option | script.nsi | -] [...]

3.1.1 Options

- /LICENSE displays license information.
- The /V switch followed by a number between 0 and 4 will set the verbosity of output accordingly. 0=no output, 1=errors only, 2=warnings and errors, 3=info, warnings, and errors, 4=all output.
- The /P switch followed by a number between 0 and 5 will set the priority of the compiler process accordingly. 0=idle, 1=below normal, 2=normal (default), 3=above normal, 4=high, 5=realtime.
- The /O switch followed by a filename tells the compiler to print its log to that file (instead of the screen)
- /PAUSE makes makensis pause before quitting, which is useful when executing directly from Windows.
- /NOCONFIG disables inclusion of nsisconf.nsh. Without this parameter, installer defaults are set from nsisconf.nsh.
- /CMDHELP prints basic usage information for command (if specified), or all commands (if command is not specified).
- /HDRINFO prints information about which options were used to compile makensis.
- /NOCD disables the current directory change to that of the .nsi file
- /INPUTCHARSET allows you to specify a specific codepage for files without a BOM. (ACP|OEM|CP#|UTF8|UTF16<LE|BE>)
- /OUTPUTCHARSET allows you to specify the codepage used by stdout when the output is redirected.

(ACP|OEM|CP#|UTF8[SIG]|UTF16<LE|BE>[BOM])

- /PPO or /SAFEPPO will only run the preprocessor and print the result to stdout. The safe version will not execute instructions like !appendfile or !system. !packhdr and !finalize are never executed.
- /WX treats warnings as errors
- Using the /D switch one or more times will add to symbols to the globally defined list (See !define).
- Using the /X switch one or more times will execute the code you specify following it. Example: "/XAutoCloseWindow false"
- Specifying a dash (-) for the script name will tell makensis to use the standard input as a source.

3.1.2 Notes

- Parameters are processed in order. makensis /Ddef script.nsi is not the same as makensis script.nsi /Ddef.
- If multiple scripts are specified, they are treated as one concatenated script.

3.1.3 Environment variables

makensis checks a number of environment variables that tell it where to locate the things it needs in order to create installers. These variables include:

- NSISDIR, NSISCONFDIR Places where nsis data and config files are installed. NSISDIR alters the script variable \${NSISDIR}. See section 4.2.3 for more info.
- APPDATA (on Windows) or HOME (on other platforms) Location of the per-user configuration file.

3.1.4 Examples

Basic usage:

makensis.exe myscript.nsi

Quiet mode:

makensis.exe /V1 myscript.nsi

Force compressor:

makensis.exe /X"SetCompressor /FINAL lzma" myscript.ns Change script behavior:

makensis.exe /DUSE_UPX /DVERSION=1.337 /DNO_IMAGES mys Parameters order:

makensis /XSection sectioncontents.nsi /XSectionEnd

Previous | Contents | Next

3.2 Installer Usage

Generated installers and uninstallers accept a few options on the command line. These options give the user a bit more control over the installation process.

3.2.1 Common Options

- /NCRC disables the CRC check, unless crccheck force was used in the script.
- /S runs the installer or uninstaller silently. See section 4.12 for more information.
- /D sets the default installation directory (\$INSTDIR), overriding InstallDir and InstallDirRegKey. It must be the last parameter used in the command line and must not contain any quotes, even if the path contains spaces. Only absolute paths are supported.

3.2.2 Uninstaller Specific Options

 _?= sets \$INSTDIR. It also stops the uninstaller from copying itself to the temporary directory and running from there. It can be used along with ExecWait to wait for the uninstaller to finish. It must be the last parameter used in the command line and must not contain any quotes, even if the path contains spaces.

3.2.3 Examples

```
installer.exe /NCRC
installer.exe /S
installer.exe /D=C:\Program Files\NSIS
installer.exe /NCRC /S /D=C:\Program Files\NSIS
uninstaller.exe /S ?=C:\Program Files\NSIS
```

uninstall old version
ExecWait '"\$INSTDIR\uninstaller.exe" /S _?=\$INSTDIR'

4.1 Script File Format

A NSIS Script File (.nsi) is just a text file with script code.

Commands

Commands lines are in the format 'command [parameters]'

```
File "myfile"
```

Comments

Lines beginning with; or # are comments. You can put comments after commands. You can also use C-style comments to comment one or more lines.

If you want a parameter to start with; or # put it in quotes.

Plug-ins

To call a plug-in, use 'plugin::command [parameters]'. For more info see Plug-in DLLs.

nsExec::Exec "myfile"

Numbers

For parameters that are treated as numbers, use decimal (the number) or hexadecimal (with 0x prepended to it, i.e. 0x12345AB), or octal (numbers beginning with a 0 and no x).

Colors should be set in hexadecimal RGB format, like HTML but without the #.

```
IntCmp 1 0x1 lbl_equal
SetCtlColors $HWND CCCCCC
```

Strings

To represent strings that have spaces, use quotes:

```
MessageBox MB_OK "Hi there!"
```

Quotes only have the property of containing a parameter if they surround the rest of the parameter. They can be either single quotes, double quotes, or the backward single quote.

You can escape quotes using \$\:

```
MessageBox MB_OK "I'll be happy"; this one puts a ' i MessageBox MB_OK 'And he said to me "Hi there!"'; thi MessageBox MB_OK `And he said to me "I'll be happy!"` MessageBox MB_OK "$\"A quote from a wise man$\" said t
```

It is also possible to put newlines, tabs etc. in a string using \$\r, \$\n, \$\t

Variables

Variables start with \$. User variables must be declared.

```
Var MYVAR
StrCpy $MYVAR "myvalue"
```

More information...

Long commands

To extend a command over multiple lines, use a backslash (\) at the end of the line. The next line will effectively be concatenated to the end of it. For example:

```
CreateShortcut "$SMPROGRAMS\NSIS\ZIP2EXE project works
    "$INSTDIR\source\zip2exe\zip2exe.dsw"

MessageBox MB_YESNO|MB_ICONQUESTION \
    "Do you want to remove all files in the folder? \
    (If you have anything you created that you want \
        to keep, click No)" \
    IDNO NoRemoveLabel
```

Line extension for long commands works for comments as well. It can be a bit confusing, so it should be avoided.

```
# A comment \
still a comment here...
```

Configuration file

If a file named "nsisconf.nsh" in the config directory exists, it will be included by default before any scripts (unless the /NOCONFIG command line parameter is used). The config directory on Windows is the same directory as makensis.exe is in. On other platforms this is set at install time and defaults to \$PREFIX/etc/. You can alter this at runtime, see section 3.1.3 for more information.

4.2 Variables

All variables are global and can be used in Sections or Functions. Note that by default, variables are limited to 1024 characters. To extend this limit, build NSIS with a bigger value of the NSIS_MAX_STRLEN build setting or use the special build.

4.2.1 User Variables

\$VARNAME

User variables must be declared with the Var command. You can use these variables to store values, work with string manipulation etc.

4.2.1.1 Var

[/GLOBAL] var name

Declare a user variable. Allowed characters for variables names: [a-z][A-Z][0-9] and '_'. All defined variables are global, even if defined in a section or a function. To make this clear, variables defined in a section or a function must use the /GLOBAL flag. The /GLOBAL flag is not required outside of sections and functions.

```
Var example

Function testVar
   Var /GLOBAL example2

StrCpy $example "example value"
   StrCpy $example2 "another example value"
FunctionEnd
```

4.2.2 Other Writable Variables

\$0, \$1, \$2, \$3, \$4, \$5, \$6, \$7, \$8, \$9, \$R0, \$R1, \$R2, \$R3, \$R4, \$R5, \$R6, \$R7, \$R8, \$R9

Registers. These variables can be used just like user variables, but are usually used in shared functions or macros. You don't have to declare these variables so you won't get any name conflicts when using them in shared code. When using these variables in shared code it's recommended that you use the stack to save and restore their original values. These variables can also be used for communication with plugins because they can be read and written by the plug-in DLLs.

\$INSTDIR

The installation directory (\$INSTDIR is modifiable using StrCpy, ReadRegStr, ReadINIStr, etc. - This could be used, for example, in the .onInit function to do a more advanced detection of install location).

Note that in uninstaller code, \$INSTDIR contains the directory where the uninstaller lies. It does **not** necessarily contain the same value it contained in the installer. For example, if you write the uninstaller to \$WINDIR and the user doesn't move it, \$INSTDIR will be \$WINDIR in the uninstaller. If you write the uninstaller to another location, you should keep the installer's \$INSTDIR in the registry or an alternative storing facility and read it in the uninstaller.

\$OUTDIR

The current output directory (set implicitly via SetOutPath or explicitly via StrCpy, ReadRegStr, ReadINIStr, etc)

\$CMDLINE

The command line of the installer. The format of the command line can be one of the following:

- "full\path to\installer.exe" PARAMETER PARAMETER PARAMETER
- installer.exe PARAMETER PARAMETER
- For parsing out the PARAMETER portion, see GetParameters. If /D=
 is specified on the command line (to override the install directory) it

won't show up in \$CMDLINE.

\$LANGUAGE

The identifier of the language that is currently used. For example, English is 1033. You can only change this variable in .onlnit.

4.2.3 Constants

Constants can also be used in the InstallDir attribute.

Note that some of the new constants will not work on every OS. For example, \$CDBURN_AREA will only work on Windows XP and above. If it's used on Windows 98, it'll be empty. Unless mentioned otherwise, a constant should be available on every OS.

\$PROGRAMFILES, \$PROGRAMFILES32, \$PROGRAMFILES64

The program files directory (usually c:\Program Files but detected at runtime). On Windows x64, \$PROGRAMFILES and \$PROGRAMFILES32 point to c:\Program Files (x86) while \$PROGRAMFILES64 points to c:\Program Files. Use \$PROGRAMFILES64 when installing x64 applications.

\$COMMONFILES, \$COMMONFILES32, \$COMMONFILES64

The common files directory. This is a directory for components that are shared across applications (usually c:\Program Files\Common Files but detected at runtime). On Windows x64, \$COMMONFILES and \$COMMONFILES32 point to c:\Program Files (x86)\Common Files while \$COMMONFILES64 points to c:\Program Files\Common Files. Use \$COMMONFILES64 when installing x64 applications.

\$DESKTOP

The Windows desktop directory. The context of this constant (All Users or Current user) depends on the SetShellVarContext setting. The default is the current user.

\$EXEDIR

The directory containing the installer executable (technically this is a variable and you can modify it, but it is probably not a good idea).

\$EXEFILE

The base name of the installer executable.

\$EXEPATH

The full path of the installer executable.

\${NSISDIR}

A symbol that contains the path where NSIS is installed. Useful if you want to reference resources that are in NSIS directory e.g. Icons, UIs etc.

When compiled with support for keeping makensis and the data in the same place (the default on Windows), it is in the same place as makensis, on other platforms it is set at compile time (See the INSTALL file for info). In both instances you can modify it at runtime by setting the NSISDIR environment variable. See section 3.1.3 for more info.

\$WINDIR

The Windows directory (usually c:\windows or c:\winnt but detected at runtime).

\$SYSDIR

The Windows system directory (usually c:\windows\System or C:\winNT\System32 but detected at runtime).

\$TEMP

The temporary directory.

\$STARTMENU

The start menu folder (useful for adding start menu items using CreateShortcut). The context of this constant (All Users or Current user) depends on the SetShellVarContext setting. The default is the current user.

\$SMPROGRAMS

The start menu programs folder (use this whenever you want \$STARTMENU\Programs). The context of this constant (All Users or Current user) depends on the SetShellVarContext setting. The default is the current user.

\$SMSTARTUP

The start menu programs / startup folder. The context of this constant (All Users or Current user) depends on the SetShellVarContext setting. The default is the current user.

\$QUICKLAUNCH

The quick launch folder for IE4 active desktop and above. If quick launch is not available it simply returns the same as \$TEMP.

\$DOCUMENTS

The documents directory. A typical path for the current user is C:\Documents and Settings\Foo\My Documents. The context of this constant (All Users or Current user) depends on the SetShellVarContext setting. The default is the current user.

This constant is not available on Windows 95 unless Internet Explorer 4 is installed.

\$SENDTO

The directory that contains Send To menu shortcut items.

\$RECENT

The directory that contains shortcuts to the user's recently used documents.

\$FAVORITES

The directory that contains shortcuts to the user's favorite websites, documents, etc. The context of this constant (All Users or Current user) depends on the SetShellVarContext setting. The default is the current user.

This constant is not available on Windows 95 unless Internet Explorer 4 is installed.

\$MUSIC

The user's music files directory. The context of this constant (All Users or Current user) depends on the SetShellVarContext setting. The default is the current user.

This constant is available on Windows XP, ME and above.

\$PICTURES

The user's picture files directory. The context of this constant (All Users or Current user) depends on the SetShellVarContext setting. The default is the current user.

This constant is available on Windows 2000, XP, ME and above.

\$VIDEOS

The user's video files directory. The context of this constant (All Users or Current user) depends on the SetShellVarContext setting. The default is the current user.

This constant is available on Windows XP, ME and above.

\$NETHOOD

The directory that contains link objects that may exist in the My Network Places/Network Neighborhood folder.

This constant is not available on Windows 95 unless Internet Explorer 4 with Active Desktop is installed.

\$FONTS

The system's fonts directory.

\$TEMPLATES

The document templates directory. The context of this constant (All Users or Current user) depends on the SetShellVarContext setting. The default is the current user.

\$APPDATA

The application data directory. Detection of the current user path requires Internet Explorer 4 and above. Detection of the all users path requires Internet Explorer 5 and above. The context of this constant (All Users or Current user) depends on the SetShellVarContext setting. The default is the current user.

This constant is not available on Windows 95 unless Internet Explorer 4 with Active Desktop is installed.

\$LOCALAPPDATA

The local (non-roaming) application data directory.

This constant is available on Windows 2000 and above.

\$PRINTHOOD

The directory that contains link objects that may exist in the Printers folder.

This constant is not available on Windows 95 and Windows 98.

\$INTERNET_CACHE

Internet Explorer's temporary internet files directory.

This constant is not available on Windows 95 nor Windows NT 4 unless Internet Explorer 4 with Active Desktop is installed.

\$COOKIES

Internet Explorer's cookies directory.

This constant is not available on Windows 95 nor Windows NT 4 unless Internet Explorer 4 with Active Desktop is installed.

\$HISTORY

Internet Explorer's history directory.

This constant is not available on Windows 95 nor Windows NT 4 unless Internet Explorer 4 with Active Desktop is installed.

\$PROFILE

The user's profile directory. A typical path is C:\Documents and Settings\Foo.

This constant is available on Windows 2000 and above.

\$ADMINTOOLS

A directory where administrative tools are kept. The context of this constant (All Users or Current user) depends on the SetShellVarContext setting. The default is the current user.

This constant is available on Windows 2000, ME and above.

\$RESOURCES

The resources directory that stores themes and other Windows resources (usually \$windiresources but detected at runtime).

This constant is available on Windows XP and above.

\$RESOURCES_LOCALIZED

The localized resources directory that stores themes and other Windows resources (usually \$windir\Resources\1033 but detected at runtime).

This constant is available on Windows XP and above.

\$CDBURN AREA

A directory where files awaiting to be burned to CD are stored.

This constant is available on Windows XP and above.

\$HWNDPARENT

HWND of the main window (in decimal).

\$PLUGINSDIR

The path to a temporary folder created upon the first usage of a plug-in or a call to InitPluginsDir. This folder is automatically deleted when the installer exits. This makes this folder the ideal folder to hold INI files for InstallOptions, bitmaps for the splash plug-in, or any other file that a plug-in needs to work.

4.2.4 Constants Used in Strings

\$\$

Use to represent \$.

\$\r

Use to represent a carriage return (\r).

\$\n

Use to represent a newline (\n).

\$\t

Use to represent a tab (\t).

4.3 Labels

Labels are the targets of Goto instructions and the various branching instructions (such as IfErrors, MessageBox, IfFileExists, and StrCmp). Labels must be within a Section or a Function. Labels are local in scope, meaning they are only accessible from within the Section or Function that they reside in. To declare a label, simply use:

MyLabel:

Labels cannot begin with a -, +, !, \$, or 0-9. When specifying labels for the various instructions that require them, remember that both an empty string ("") and 0 both represent the next instruction (meaning no Goto will occur). Labels beginning with a period (.) are global, meaning you can jump to them from any function or section (though you cannot jump to an uninstall global label from the installer, and vice versa).

4.4 Relative Jumps

Unlike labels, relative jumps are, as the name suggests, relative to the place they are called from. You can use relative jumps wherever you can use labels. Relative jumps are marked by numbers. +1 jumps to the next instruction (the default advancement), +2 will skip one instruction and go to the second instruction from the current instruction, -2 will jump two instructions backward, and +10 will skip 9 instructions, jumping to the tenth instruction from the current instruction.

A instruction is every command that is executed at run-time, when the installer is running. MessageBox, Goto, GetDLLVersion, FileRead, SetShellVarContext are all instructions. AddSize, Section, SectionGroup, SectionEnd, SetOverwrite (and everything under Compiler Flags), Name, SetFont, LangString, are not instructions because they are executed at compile time.

Examples:

Goto +2

MessageBox MB_OK "You will never ever see this mess MessageBox MB OK "The last message was skipped, this

Goto +4
MessageBox MB_OK "The following message will be skipp Goto +3
MessageBox MB_OK "You will never ever see this messag Goto -3
MessageBox MB_OK "Done"

Note that macro insertion is not considered as one instruction when it comes to relative jumps. The macro is expanded before relative jumps are applied, and so relative jumps can jump into code inside an inserted macro. The following code, for example, will not skip the macro. It will show a message box.

```
!macro relative_jump_test
   MessageBox MB_OK "first macro line"
   MessageBox MB_OK "second macro line"
!macroend

Goto +2
!insertmacro relative_jump_test
```

4.5 Pages

Each (non-silent) NSIS installer has a set of pages. Each page can be a NSIS built-in page or a custom page created by a user's function (with nsDialogs or InstallOptions for example).

The script controls the page order, appearance, and behavior. You can skip pages, paint them white, force the user to stay in a certain page until a certain condition is met, show a readme page, show custom designed pages for input and more. In this section you will learn how to do all of the above.

There are two basic commands regarding pages, Page and UninstPage. The first adds a page to the installer, the second adds a page to the uninstaller. On top of those two there is the PageEx command which allows you to add a page to either one and with greater amount of options. PageEx allows you to set options to the specific page you are adding instead of using the default that's set outside of PageEx.

4.5.1 Ordering

The page order is set simply by the order Page, UninstPage and PageEx appear in the script. For example:

```
Page license
Page components
Page directory
Page instfiles
UninstPage uninstConfirm
UninstPage instfiles
```

This code will tell NSIS to first show the license page, then the components selection page, then the directory selection page and finally the install log where sections are executed. The uninstaller will first show the uninstall confirmation page and then the uninstallation log.

You can specify the same page type more than once.

For backwards compatibility with old NSIS scripts, the following installer pages will be added if no installer page commands are used: license (if LicenseText and LicenseData were specified), components (if ComponentText was specified and there is more than one visible section), directory (if DirText was specified) and instfiles. When there are no uninstaller page commands the following uninstaller pages will be added: uninstall confirmation page (if UninstallText was specified) and instfiles. This method is deprecated, converting scripts to use page commands is highly recommended because you can use the new standard language strings.

4.5.2 Page Options

Each page has its unique set of data that defines how it will look and act. This section describes what data each type of page uses and how you can set it. Callback functions are described below and are not dealt with in this section.

The list below lists the commands that affect a certain page type. Unless otherwise mentioned, these commands can be used both inside and outside of a PageEx block. If used inside a PageEx block they will only affect the current page being set by PageEx, otherwise they will set the default for all other pages.

License page

- LicenseText
- LicenseData
- LicenseForceSelection

Components selection page

ComponentText

Directory selection page

- DirText
- DirVar (can only be used in PageEx)
- DirVerify

Un/Installation log page

- DetailsButtonText
- CompletedText

Uninstall confirmation page

- DirVar (can only be used in PageEx)
- UninstallText

Use Caption to set the page caption.

4.5.3 Callbacks

Each built-in page has three callback functions: the pre-function, the show function and the leave-function. The pre-function is called right before the page is created, the show-function is called right after it has been created but before it is shown and the leave-function is called right after the user has pressed the next button (before actually leaving the page).

- The pre-function allows you to skip the page using Abort.
- The show-function allows you to tweak the page's user interface with CreateFont, SetCtlColors, SendMessage etc.
- The leave-function allows you to force the user to stay on the current page using Abort.

A custom page only has two callback functions, one that creates it which is mandatory, and one leave-function that acts just like the leave-function for built-in pages.

Examples:

Page license skipLicense "" stayInLicense

```
Page custom customPage "" ": custom page"
Page instfiles
Function skipLicense
 MessageBox MB YESNO "Do you want to skip the licens
   Abort
  no:
FunctionEnd
Function stayInLicense
 MessageBox MB YESNO "Do you want to stay in the lic
    Abort
  no:
FunctionEnd
Function customPage
 GetTempFileName $R0
 File /oname=$R0 customPage.ini
 InstallOptions::dialog $R0
 Pop $R1
 StrCmp $R1 "cancel" done
 StrCmp $R1 "back" done
 StrCmp $R1 "success" done
 error: MessageBox MB OK|MB ICONSTOP "InstallOptions
 done:
FunctionEnd
```

4.5.4 Page

```
custom [creator_function] [leave_function] [caption] [
   OR
internal_page_type [pre_function] [show_function] [lea
```

Adds an installer page. See the above sections for more information about built-in versus custom pages and about callback functions.

internal_page_type can be:

- license license page
- components components selection page
- *directory* installation directory selection page
- instfiles installation page where the sections are executed
- uninstConfirm uninstall confirmation page

The last page of the installer has its cancel button disabled to prevent confusion. To enable it anyway, use /ENABLECANCEL.

4.5.5 UninstPage

```
custom [creator_function] [leave_function] [caption] [
   OR
internal_page_type [pre_function] [show_function] [lea
```

Adds an uninstaller page. See the above sections for more information about built-in versus custom pages and about callback functions.

See Page for possible values of *internal_page_type*.

4.5.6 PageEx

[un.](custom|uninstConfirm|license|components|director

Adds an installer page or an uninstaller page if the un. prefix was used. Every PageEx must have a matching PageExEnd. In a PageEx block you can set options that are specific to this page and will not be used for other pages. Options that are not set will revert to what was set outside the PageEx block or the default if nothing was set. To set the sub-caption for a page use Caption or SubCaption to set the default. To set the callback functions for a page set with PageEx use PageCallbacks. See the above sections for more information about built-in versus custom pages.

Example usage:

```
PageEx license
LicenseText "Readme"
```

```
LicenseData readme.rtf
PageExEnd

PageEx license
  LicenseData license.txt
  LicenseForceSelection checkbox
PageExEnd
```

4.5.7 PageExEnd

Ends a PageEx block.

4.5.8 PageCallbacks

```
([creator_function] [leave_function]) | ([pre_function
```

Sets the callback functions for a page defined using PageEx. Can only be used inside a PageEx block. See the above sections for more information about callback functions.

PageEx license PageCallbacks licensePre licenseShow licenseLeave PageExEnd

4.6 Sections

Each NSIS installer contains one or more sections. Each of these sections are created, modified, and ended with the following commands.

- Each section contains zero or more instructions.
- Sections are executed in order by the resulting installer, and if a component page is used, the user will have the option of disabling/enabling each visible section.
- If a section's name is 'Uninstall' or is prefixed with 'un.', it's an uninstaller section.

4.6.1 Section Commands

4.6.1.1 AddSize

size_kb

Tells the installer that the current section needs an additional "size_kb" kilobytes of disk space. Only valid within a section (will have no effect outside of a section or in a function).

Section AddSize 500 SectionEnd

4.6.1.2 Section

[/o] [([!]|[-])section name] [section index output]

Begins and opens a new section. If section_name is empty, omitted, or begins with a -, then it is a hidden section and the user will not have the option of disabling it. If the section name is 'Uninstall' or is prefixed with 'un.', then it is a an uninstaller section. If section_index_output is specified, the parameter will be !defined with the section index (can be

used with SectionSetText etc). If the section name begins with a !, the section will be displayed as bold. If the /o switch is specified, the section will be unselected by default.

```
Section "-hidden section"
SectionEnd

Section # hidden section
SectionEnd

Section "!bold section"
SectionEnd

SectionEnd

Section /o "optional"
SectionEnd

SectionEnd

Section "install something" SEC_IDX
SectionEnd
```

To access the section index, curly brackets must be used and the code must be located below the section in the script.

```
Section test1 sec1_id
SectionEnd

Section test2 sec2_id
SectionEnd

Function .onInit
   SectionGetText ${sec2_id} $0
   MessageBox MB_OK "name of ${sec2_id}:$\n$0" # will c
FunctionEnd

Function .onInit
   SectionGetText ${sec2_id} $0
   MessageBox MB_OK "name of ${sec2_id}:$\n$0" # will i
        # plus a warning stating:
```

```
# unknown variable/constant "{sec2_id}" detected
FunctionEnd

Section test1 sec1_id
SectionEnd

Section test2 sec2_id
SectionEnd
```

4.6.1.3 SectionEnd

This command closes the current open section.

4.6.1.4 SectionIn

```
insttype_index [insttype_index] [R0]
```

This command specifies which install types (see InstType) the current section defaults to the enabled state in. Multiple SectionIn commands can be specified (they are combined). If you specify RO as a parameter, then the section will be read-only, meaning the user won't be able to change its state. The first install type defined using InstType is indexed 1, the next 2 and so on.

```
InstType "full"
InstType "minimal"

Section "a section"
SectionEnd

Section "another section"
SectionIn 1
SectionEnd
```

4.6.1.5 SectionGroup

[/e] section_group_name [index_output]

This command inserts a section group. The section group must be closed with SectionGroupEnd, and should contain 1 or more sections. If the section group name begins with a!, its name will be displayed with a bold font. If /e is present, the section group will be expanded by default. If <code>index_output</code> is specified, the parameter will be !defined with the section index (can be used with SectionSetText etc). If the name is prefixed with 'un.' the section group is an uninstaller section group.

```
SectionGroup "some stuff"
Section "a section"
SectionEnd
Section "another section"
SectionEnd
SectionEnd
SectionGroupEnd
```

4.6.1.6 SectionGroupEnd

Closes a section group opened with SectionGroup.

4.6.2 Uninstall Section

A special Section named 'Uninstall' must be created in order to generate an uninstaller. This section should remove all files, registry keys etc etc that were installed by the installer, from the system. Here is an example of a simple uninstall section:

```
Section "Uninstall"
  Delete $INSTDIR\Uninst.exe ; delete self (see explan
  Delete $INSTDIR\myApp.exe
  RMDir $INSTDIR
  DeleteRegKey HKLM SOFTWARE\myApp
SectionEnd
```

The first Delete instruction works (deleting the uninstaller), because the uninstaller is transparently copied to the system temporary directory for

the uninstall.

Note that in uninstaller code, \$INSTDIR contains the directory where the uninstaller lies. It does **not** necessarily contain the same value it contained in the installer.

Previous | Contents | Next

Previous | Contents | Next

4.7 Functions

Functions are similar to Sections in that they contain zero or more instructions. User functions are not called by the installer directly, instead they are called from Sections using the Call instruction. Callback functions will be called by the installer when a certain event occurs.

Functions must be declared outside of Sections or other Functions.

4.7.1 Function Commands

4.7.1.1 Function

[function_name]

Begins and opens a new function. Function names beginning with "." (e.g. ".Whatever") are generally reserved for callback functions. Function names beginning with "un." are functions that will be generated in the Uninstaller. Hence, normal install Sections and functions cannot call uninstall functions, and the Uninstall Section and uninstall functions cannot call normal functions.

Function func
 # some commands
FunctionEnd

Section
Call func
SectionEnd

4.7.1.2 FunctionEnd

This command closes the current open function.

4.7.2 Callback Functions

You can create callback functions (which have special names), that will be called by the installer at certain points in the install. Below is a list of available callbacks:

4.7.2.1 Install Callbacks

4.7.2.1.1 .onGUIInit

This callback will be called just before the first page is loaded and the installer dialog is shown, allowing you to tweak the user interface.

Example:

```
!include "WinMessages.nsh"

Function .onGUIInit
  # 1028 is the id of the branding text control
  GetDlgItem $R0 $HWNDPARENT 1028
  CreateFont $R1 "Tahoma" 10 700
  SendMessage $R0 ${WM_SETFONT} $R1 0
  # set background color to white and text color to r
  SetCtlColors $R0 FFFFFF FF0000
FunctionEnd
```

4.7.2.1.2 .onInit

This callback will be called when the installer is nearly finished initializing. If the '.onInit' function calls Abort, the installer will quit instantly.

Here are two examples of how this might be used:

```
Function .onInit
   MessageBox MB_YESNO "This will install. Continue?"
   Abort ; causes installer to quit.
   NoAbort:
FunctionEnd
```

```
Function .onInit
  ReadINIStr $INSTDIR $WINDIR\wincmd.ini Configuratio
  StrCmp $INSTDIR "" 0 NoAbort
    MessageBox MB_OK "Windows Commander not found. Un
    Abort ; causes installer to quit.
    NoAbort:
FunctionEnd
```

4.7.2.1.3 .onInstFailed

This callback is called when the user hits the 'cancel' button after the install has failed (if it could not extract a file, or the install script used the Abort command).

Example:

```
Function .onInstFailed
  MessageBox MB_OK "Better luck next time."
FunctionEnd
```

4.7.2.1.4 .onInstSuccess

This callback is called when the install was successful, right before the install window closes (which may be after the user clicks 'Close' if AutoCloseWindow or SetAutoClose is set to false).

Example:

```
Function .onInstSuccess
  MessageBox MB_YESNO "Congrats, it worked. View rea
     Exec notepad.exe ; view readme or whatever, if y
  NoReadme:
FunctionEnd
```

4.7.2.1.5 .onGUIEnd

This callback is called right after the installer window closes. Use it to free any user interface related plug-ins if needed.

4.7.2.1.6 .onMouseOverSection

This callback is called whenever the mouse position over the sections tree has changed. This allows you to set a description for each section for example. The section id on which the mouse is over currently is stored, temporarily, in \$0.

Example:

```
Function .onMouseOverSection
  FindWindow $R0 "#32770" "" $HWNDPARENT
  GetDlgItem $R0 $R0 1043; description item (must b
  StrCmp $0 0 "" +2
    SendMessage $R0 ${WM_SETTEXT} 0 "STR:first secti
  StrCmp $0 1 "" +2
    SendMessage $R0 ${WM_SETTEXT} 0 "STR:second sect
  FunctionEnd
```

4.7.2.1.7 .onRebootFailed

This callback is called if Reboot fails. WriteUninstaller, plug-ins, File and WriteRegBin should not be used in this callback.

Example:

```
Function .onRebootFailed
  MessageBox MB_OK|MB_ICONSTOP "Reboot failed. Please
FunctionEnd
```

4.7.2.1.8 .onSelChange

Called when the selection changes on the component page. Useful for using with SectionSetFlags and SectionGetFlags.

Selection changes include both section selection and installation type changes. The section id of the changed section is stored in \$0. \$0 is -1 if the installation type changed. You only get notifications for changes initiated by the user and only one notification per action even if the action also affected child sections and/or parent groups.

4.7.2.1.9 .onUserAbort

This callback is called when the user hits the 'cancel' button, and the install hasn't already failed. If this function calls Abort, the install will not be aborted.

Example:

```
Function .onUserAbort
  MessageBox MB_YESNO "Abort install?" IDYES NoCancel
   Abort ; causes installer to not quit.
  NoCancelAbort:
FunctionEnd
```

4.7.2.1.10 .onVerifyInstDir

This callback enables control over whether or not an installation path is valid for your installer. This code will be called every time the user changes the install directory, so it shouldn't do anything crazy with MessageBox or the like. If this function calls Abort, the installation path in \$INSTDIR is deemed invalid.

Example:

```
Function .onVerifyInstDir
   IfFileExists $INSTDIR\Winamp.exe PathGood
   Abort ; if $INSTDIR is not a winamp directory, d
   PathGood:
FunctionEnd
```

4.7.2.2 Uninstall Callbacks

4.7.2.2.1 un.onGUIInit

This callback will be called just before the first page is loaded and the installer dialog is shown, allowing you to tweak the user interface.

Have a look at .onGUIInit for an example.

4.7.2.2.2 un.onInit

This callback will be called when the uninstaller is nearly finished initializing. If the 'un.onInit' function calls Abort, the uninstaller will quit instantly. Note that this function can verify and/or modify \$INSTDIR if necessary.

Here are two examples of how this might be used:

```
Function un.onInit
   MessageBox MB_YESNO "This will uninstall. Continue
   Abort ; causes uninstaller to quit.
   NoAbort:
   FunctionEnd
```

or:

```
Function un.onInit
   IfFileExists $INSTDIR\myfile.exe found
     Messagebox MB_OK "Uninstall path incorrect"
   Abort
   found:
FunctionEnd
```

4.7.2.2.3 un.onUninstFailed

This callback is called when the user hits the 'cancel' button after the uninstall has failed (if it used the Abort command or otherwise failed).

Example:

```
Function un.onUninstFailed
  MessageBox MB_OK "Better luck next time."
FunctionEnd
```

4.7.2.2.4 un.onUninstSuccess

This callback is called when the uninstall was successful, right before the install window closes (which may be after the user clicks 'Close' if SetAutoClose is set to false)..

Example:

```
Function un.onUninstSuccess
   MessageBox MB_OK "Congrats, it's gone."
FunctionEnd
```

4.7.2.2.5 un.onGUIEnd

This callback is called right after the uninstaller window closes. Use it to free any user interface related plug-ins if needed.

4.7.2.2.6 un.onRebootFailed

This callback is called if Reboot fails. WriteUninstaller, plug-ins, File and WriteRegBin should not be used in this callback.

Example:

```
Function un.onRebootFailed
  MessageBox MB_OK|MB_ICONSTOP "Reboot failed. Please
FunctionEnd
```

4.7.2.2.7 un.onSelChange

Called when the selection changes on the component page. Useful for using with SectionSetFlags and SectionGetFlags.

Selection changes include both section selection and installation type

changes. The section id of the changed section is stored in \$0. \$0 is -1 if the installation type changed. You only get notifications for changes initiated by the user and only one notification per action even if the action also affected child sections and/or parent groups.

4.7.2.2.8 un.onUserAbort

This callback is called when the user hits the 'cancel' button and the uninstall hasn't already failed. If this function calls Abort, the install will not be aborted.

Example:

Function un.onUserAbort
 MessageBox MB_YESNO "Abort uninstall?" IDYES NoCan
 Abort ; causes uninstaller to not quit.
 NoCancelAbort:
FunctionEnd

Previous | Contents | Next

Previous | Contents | Next

4.8 Installer Attributes

4.8.1 General Attributes

The commands below all adjust attributes of the installer. These attributes control how the installer looks and functions, including which pages are present in the installer, which text is displayed in each part of each page, the name of the installer, the icon the installer uses, the default installation directory and more. Note that these attributes can be set anywhere in the file except in a Section or Function.

Defaults are bold and underlined

4.8.1.1 AddBrandingImage

```
(left|right|top|bottom) (width|height) [padding]
```

Adds a branding image on the top, bottom, left, or right of the installer. Its size will be set according to the width/height specified, the installer width/height and the installers font. The final size will not always be what you requested; have a look at the output of the command for the actual size. Because this depends on the installers font, you should use SetFont before AddBrandingImage. The default padding value is 2.

AddBrandingImage only adds a placeholder for an image. To set the image itself at runtime, use SetBrandingImage.

```
AddBrandingImage left 100
AddBrandingImage right 50
AddBrandingImage top 20
AddBrandingImage bottom 35
AddBrandingImage left 100 5
```

4.8.1.2 AllowRootDirInstall

true| false

Controls whether or not installs are allowed in the root directory of a drive, or directly into a network share. Set to 'true' to change the safe behavior, which prevents users from selecting C:\ or \\Server\Share as an install (and later on, uninstall) directory. For additional directory selection page customizability, see .on\VerifyInstDir.

4.8.1.3 AutoCloseWindow

true| false

Sets whether or not the install window automatically closes when completed. This is overrideable from a section using SetAutoClose.

4.8.1.4 BGFont

[font_face [height [weight] [/ITALIC] [/UNDERLINE] [/S

Specifies the font used to show the text on the background gradient. To set the color use BGGradient. The default font will be used if no parameters are specified. The default font is bold and italic Times New Roman.

4.8.1.5 BGGradient

[off|(topc botc [textcolor|notext])]

Specifies whether or not to use a gradient background window. If 'off', the installer will not show a background window, if no parameters are specified, the default black to blue gradient is used, and otherwise the top_color or bottom_color are used to make a gradient. Top_color and bottom_color are specified using the form RRGGBB (in hexadecimal, as in HTML, only minus the leading '#', since # can be used for comments). 'textcolor' can be specified as well, or 'notext' can be specified to turn the

big background text off.

4.8.1.6 BrandingText

/TRIM(LEFT|RIGHT|CENTER) text

Sets the text that is shown at the bottom of the install window (by default it is 'Nullsoft Install System vX.XX'). Setting this to an empty string ("") uses the default; to set the string to blank, use " " (a space). If it doesn't matter to you, leave it the default so that everybody can know why the installer didn't suck :). Use /TRIMLEFT, /TRIMRIGHT or /TRIMCENTER to trim down the size of the control to the size of the string.

Accepts variables. If variables are used, they must be initialized on onlnit.

4.8.1.7 Caption

caption

When used outside a PageEx block: Sets the text for the titlebar of the installer. By default it is '\$(^Name) Setup', where Name is specified by the Name instruction. You can however override it with 'MyApp Installer' or whatever. If you specify an empty string (""), the default will be used (you can specify " " to simulate a empty string).

When used inside a PageEx block: Sets the subcaption of the current page.

Accepts variables. If variables are used, they must be initialized on onlnit.

4.8.1.8 ChangeUI

dialog ui file.exe

Replaces dialog (*IDD_LICENSE*, *IDD_DIR*, *IDD_SELCOM*, *IDD_INST*, *IDD_INSTFILES*, *IDD_UNINST* or *IDD_VERIFY*) with a dialog from ui_file.exe with the same resource ID. You can also specify 'all' as the dialog if you wish to replace all 7 of the dialogs at once from the same UI file. For some example UIs look at Contrib\UIs under your NSIS directory.

- IDD_LICENSE must contain IDC_EDIT1 (RICHEDIT control).
- *IDD_DIR* must contain *IDC_DIR* (edit box), *IDC_BROWSE* (button) and *IDC_CHECK1* (checkbox).
- *IDD_SELCOM* must contain *IDC_TREE1* (SysTreeView32 control), and *IDC_COMBO1* (combo box).
- *IDD_INST* must contain *IDC_BACK* (button), *IDC_CHILDRECT* (static control the size of all other dialogs), *IDC_VERSTR* (static), *IDOK* (button), and *IDCANCEL* (button). If an image control (static with *SS_BITMAP* style) will be found in this dialog it will be used as the default for SetBrandingImage.
- IDD_INSTFILES must contain IDC_LIST1 (SysListView32 control), IDC_PROGRESS (msctls_progress32 control), and IDC_SHOWDETAILS (button).
- IDD_UNINST must contain IDC_EDIT1 (edit box).
- *IDD_VERIFY* must contain *IDC_STR* (static).

ChangeUI all "\${NSISDIR}\Contrib\UIs\sdbarker tiny.exe

4.8.1.9 CheckBitmap

bitmap.bmp

Specifies the bitmap with the checkbox images used in the componentselection page treeview.

This bitmap should have a size of 96x16 pixels, no more than 8bpp (256 colors) and contain six 16x16 images for the different states (in order: selection mask, not checked, checked, greyed out, unchecked & readonly, checked & readonly). Use magenta as mask color (this area will be transparent).

4.8.1.10 CompletedText

text

Replaces the default text ("Completed") that is printed at the end of the install if parameter is specified. Otherwise, the default is used.

Accepts variables. If variables are used, they must be initialized before the message is printed.

4.8.1.11 ComponentText

[text [subtext] [subtext2]]

Used to change the default text on the component page.

text: Text above the controls, to the right of the installation icon.

subtext: Text next to the installation type selection.

subtext2: Text to the left of the components list and below the installation type.

The default string will be used if a string is empty ("").

Accepts variables. If variables are used, they must be initialized before the components page is created.

4.8.1.12 CRCCheck

on|off|force

Specifies whether or not the installer will perform a CRC on itself before allowing an install. Note that if the user uses /NCRC on the command line when executing the installer, and you didn't specify 'force', the CRC will not occur, and the user will be allowed to install a (potentially) corrupted installer.

4.8.1.13 DetailsButtonText

show_details_text

Replaces the default details button text of "Show details", if parameter is specified (otherwise the default is used).

Accepts variables. If variables are used, they must be initialized before the install log (instfiles) page is created.

4.8.1.14 DirText

[text] [subtext] [browse_button_text] [browse_dlg_text

Used to change the default text on the directory page.

text: Text above the controls, to the right of the installation icon.

subtext: Text on the directory selection frame.

browse_button_text: Text on the Browse button.

browse_dlg_text: Text on the "Browse For Folder" dialog, appears after clicking on "Browse" button.

The default string will be used if a string is empty ("").

Accepts variables. If variables are used, they must be initialized before the directory page is created.

4.8.1.15 DirVar

user_var(dir input/output)

Specifies which variable is to be used to contain the directory selected. This variable should be initialized with a default value. This allows you to easily create two different directory pages that will not require you to

move values in and out of \$INSTDIR. The default variable is \$INSTDIR. This can only be used in PageEx for directory and uninstConfirm pages.

```
Var ANOTHER_DIR
PageEx directory
   DirVar $ANOTHER_DIR
PageExEnd

Section
   SetOutPath $INSTDIR
   File "a file.dat"
   SetOutPath $ANOTHER_DIR
   File "another file.dat"
SectionEnd
```

4.8.1.16 DirVerify

auto|leave

If `DirVerify leave' is used, the Next button will not be disabled if the installation directory is not valid or there is not enough space. A flag that you can read in the leave function using GetInstDirError will be set instead.

```
PageEx directory
DirVerify leave
PageCallbacks "" "" dirLeave
PageExEnd
```

4.8.1.17 FileErrorText

```
file_error_text [noignore_file_error_text]
```

Replaces the default text that comes up when a file cannot be written to. This string can contain a reference to \$0, which is the filename (\$0 is temporarily changed to this value). Example: "Can not write to file

\$\r\$\n\$0\$\r\$\ngood luck.".

Accepts variables. If variables are used, they must be initialized before File is used.

4.8.1.18 Icon

[path\]icon.ico

Sets the icon of the installer. Every image in the icon file will be included in the installer. Use UninstallIcon to set the uninstaller icon.

4.8.1.19 InstallButtonText

install button text

If parameter is specified, overrides the default install button text (of "Install") with the specified text.

Accepts variables. If variables are used, they must be initialized before the install button shows.

4.8.1.20 InstallColors

/windows | (foreground color background color)

Sets the colors to use for the install info screen (the default is 00FF00 000000. Use the form RRGGBB (in hexadecimal, as in HTML, only minus the leading '#', since # can be used for comments). Note that if "/windows" is specified as the only parameter, the default windows colors will be used.

4.8.1.21 InstallDir

definstdir

Sets the default installation directory. See the variables section for variables that can be used to make this string (especially \$PROGRAMFILES). Note that the part of this string following the last \ will be used if the user selects 'browse', and may be appended back on to the string at install time (to disable this, end the directory with a \ (which will require the entire parameter to be enclosed with quotes). If this doesn't make any sense, play around with the browse button a bit.

4.8.1.22 InstallDirRegKey

root_key subkey key_name

This attribute tells the installer to check a string in the registry and use it as the install dir if that string is valid. If this attribute is present, it will override the InstallDir attribute if the registry key is valid, otherwise it will fall back to the InstallDir value. When querying the registry, this command will automatically remove any quotes. If the string ends in ".exe", it will automatically remove the filename component of the string (i.e. if the string is "C:\Program Files\Foo\app.exe", it will know to use "C:\Program Files\Foo"). For more advanced install directory configuration, set \$INSTDIR in .onlnit.

Language strings and variables cannot be used with InstallDirRegKey.

InstallDirRegKey HKLM Software\NSIS ""
InstallDirRegKey HKLM Software\ACME\Thingy InstallLoca

4.8.1.23 InstProgressFlags

[flag [...]]

Valid values for flag are "smooth" (smooth the progress bar) or "colored" (color the progress bar with the colors set by InstallColors. Examples: "InstProgressFlags" (default old-school windows look), "InstProgressFlags smooth" (new smooth look), "InstProgressFlags smooth colored" (colored smooth look whee). Note: neither "smooth" or

"colored" work with XPStyle on when the installer runs on Windows XP with a modern theme.

4.8.1.24 InstType

install_type_name | /NOCUSTOM | /CUSTOMSTRING=str | /C

Adds an install type to the install type list, or disables the custom install type. There can be as many as 32 types, each one specifying the name of the install type. If the name is prefixed with 'un.' it is an uninstaller install type. The name can contain variables which will be processed at runtime before the components page shows. Another way of changing the InstType name during runtime is the InstTypeSetText command. The difference is that with InstTypeSetText you are saving your precious user variables. The first type is the default (generally 'Typical'). If the /NOCUSTOM switch is specified, then the "custom" install type is disabled, and the user has to choose one of the pre-defined install types. Alternatively, if the /CUSTOMSTRING switch is specified, the parameter will override the "Custom" install type text. Alternatively, if the /COMPONENTSONLYONCUSTOM flag is specified, the component list will only be shown if the "Custom" install type is selected.

Accepts variables for type names. If variables are used, they must be initialized before the components page is created.

4.8.1.25 LicenseBkColor

color | /gray | /windows

Sets the background color of the license data. Color is specified using the form RRGGBB (in hexadecimal, as in HTML, only minus the leading '#', since # can be used for comments). Default is '/gray'. You can also use the Windows OS defined color by using '/windows'.

4.8.1.26 LicenseData

licdata.(txt|rtf)

Specifies a text file or a RTF file to use for the license that the user can read. Omit this to not have a license displayed. Note that the file must be in DOS text format (\r\n). To define a multilingual license data use LicenseLangString.

If you are using a RTF file it is recommended that you edit it with WordPad and not MS Word. Using WordPad will result in a much smaller file.

Use LicenseLangString to show a different license for every language.

4.8.1.27 LicenseForceSelection

```
(checkbox [accept text] | radiobuttons [accept text] [
```

Specifies if the displayed license must be explicitly accepted or not. This can be done either by a checkbox or by radiobuttons. By default the "next button" is disabled and will only be enabled if the checkbox is enabled or the correct radio button is selected. If off is specified the "next button" is enabled by default.

```
LicenseForceSelection checkbox
LicenseForceSelection checkbox "i accept"
LicenseForceSelection radiobuttons
LicenseForceSelection radiobuttons "i accept"
LicenseForceSelection radiobuttons "i accept" "i decli
LicenseForceSelection radiobuttons "" "i decline"
LicenseForceSelection off
```

4.8.1.28 LicenseText

```
[text [button_text]]
```

Used to change the default text on the license page.

text: Text above the controls, to the right of the installation icon.

button_text: Text on the "I Agree" button.

The default string will be used if a string is empty ("").

Accepts variables. If variables are used, they must be initialized before the license page is created.

4.8.1.29 ManifestDPIAware

notset|true|false

Declare that the installer is DPI-aware. A DPI-aware application is not scaled by the DWM (DPI virtualization) so the text is never blurry. NSIS does not scale the bitmap used by the tree control on the component page and some plugins might have compatibility issues so make sure that you test your installer at different DPI settings if you select *true*.

See MSDN for more information about DPI-aware applications.

4.8.1.30 ManifestSupportedOS

none|all|WinVista|Win7|Win8|Win8.1|Win10|{GUID} [...]

Declare that the installer is compatible with the specified Windows version(s). This adds a SupportedOS entry in the compatibility section of the application manifest. The default is Win7+8+8.1+10. *none* is the default if RequestExecutionLevel is set to *none* for compatibility reasons.

Windows 8.1 and later will fake its version number if you don't declare support for that particular version. You can read more about the other changes in behavior on MSDN.

4.8.1.31 MiscButtonText

```
[back_button_text [next_button_text] [cancel_button_te
```

Replaces the default text strings for the four buttons (< Back, Next >, Cancel and Close). If parameters are omitted, the defaults are used.

Accepts variables. If variables are used, they must be initialized in .onInit.

4.8.1.32 Name

```
name [name doubled ampersands]
```

Sets the name of the installer. The name is usually simply the product name such as 'MyApp' or 'CrapSoft MyApp'. If you have one or more ampersands (&) in the name, set the second parameter to the same name, only with doubled ampersands. For example, if your product's name is "Foo & Bar", use:

```
Name "Foo & Bar" "Foo && Bar"
```

If you have ampersands in the name and use a LangString for the name, you will have to create another one with doubled ampersands to use as the second parameter.

Accepts variables. If variables are used, they must be initialized in .onInit.

4.8.1.33 OutFile

```
[path\]install.exe
```

Specifies the output file that the MakeNSIS should write the installer to. This is just the file that MakeNSIS writes, it doesn't affect the contents of the installer.

4.8.1.34 RequestExecutionLevel

none|user|highest|admin

Specifies the requested execution level for Windows Vista and higher. The value is embedded in the installer and uninstaller's XML manifest and tells Windows which privilege level the installer requires. *user* requests the user's normal level with no administrative privileges. *highest* will request the highest execution level available for the current user and will cause Windows to prompt the user to verify privilege escalation if they are a member of the administrators group. The prompt might request for the user's password. *admin*, which is also the default, requests administrator level and will cause Windows to prompt the user as well. Specifying *none* will keep the manifest empty and let Windows decide which execution level is required. Windows automatically identifies NSIS installers and decides administrator privileges are required. Because of this, *none* and *admin* have virtually the same effect.

It's recommended that every application is marked with a required execution level. Unmarked installers are subject to compatibility mode. Workarounds of this mode include automatically moving any shortcuts created in the user's start menu to all users' start menu. Installers that don't install anything into system folders nor write to the local machine registry (HKLM) should specify *user* execution level.

More information about this topic can be found on MSDN.

4.8.1.35 SetFont

```
[/LANG=lang id] font face name font size
```

Sets the installer font. Please remember that the font you choose must be present on the user's machine as well. Don't use rare fonts that only you have.

Use the /LANG switch if you wish to set a different font for each language. For example:

```
SetFont /LANG=${LANG_ENGLISH} "English Font" 9
SetFont /LANG=${LANG_FRENCH} "French Font" 10
```

There are two LangStrings named ^Font and ^FontSize which contain the font and font size for every language.

4.8.1.36 ShowInstDetails

hide|show|nevershow

Sets whether or not the details of the install are shown. Can be 'hide' to hide the details by default, allowing the user to view them, or 'show' to show them by default, or 'nevershow', to prevent the user from ever seeing them. Note that sections can override this using SetDetailsView.

4.8.1.37 ShowUninstDetails

hide|show|nevershow

Sets whether or not the details of the uninstall are shown. Can be 'hide' to hide the details by default, allowing the user to view them, or 'show' to show them by default, or 'nevershow', to prevent the user from ever seeing them. Note that sections can override this using SetDetailsView.

4.8.1.38 SilentInstall

normal|silent|silentlog

Specifies whether or not the installer should be silent. If it is 'silent' or 'silentlog', all sections that have the SF_SELECTED flag are installed quietly (you can set this flag using SectionSetFlags), with no screen output from the installer itself (the script can still display whatever it wants, use MessageBox's /SD to specify a default for silent installers). Note that if this is set to 'normal' and the user runs the installer with /S (case sensitive) on the command line, it will behave as if SilentInstall 'silent' was used. Note: see also LogSet.

See section 4.12 for more information.

4.8.1.39 SilentUnInstall

normal|silent

Specifies whether or not the uninstaller should be silent. If it is 'silent' the uninstall sections will run quietly, with no screen output from the uninstaller itself (the script can still display whatever it wants, use MessageBox's /SD to specify a default for silent uninstallers). Note that if this is set to 'normal' and the user runs the uninstaller with /S on the command line, it will behave as if SilentUnInstall 'silent' was used.

See section 4.12 for more information.

4.8.1.40 SpaceTexts

[req_text [avail_text]]

If parameters are specified, overrides the space required and space available text ("Space required: " and "Space available: " by default). If 'none' is specified as the required text no space texts will be shown.

Accepts variables. If variables are used, they must be initialized before the components page is created.

4.8.1.41 SubCaption

[page_number subcaption]

Overrides the subcaptions for each of the installer pages (0=": License Agreement",1=": Installation Options",2=": Installation Directory", 3=": Installing Files", 4=": Completed"). If you specify an empty string (""), the default will be used (you can however specify " " to achieve a blank string).

You can also set a subcaption (or override the default) using Caption inside a PageEx block.

Accepts variables. If variables are used, they must be initialized before the relevant page is created.

4.8.1.42 UninstallButtonText

text

Changes the text of the button that by default says "Uninstall" in the uninstaller. If no parameter is specified, the default text is used.

Accepts variables. If variables are used, they must be initialized before the uninstall button shows.

4.8.1.43 UninstallCaption

caption

Sets what the titlebars of the uninstaller will display. By default it is '\$(^Name) Uninstall', where Name is specified with the Name command. You can, however, override it with 'MyApp uninstaller' or whatever. If you specify an empty string (""), the default will be used (you can specify " " to simulate a empty string).

Accepts variables. If variables are used, they must be initialized in un.onlnit.

4.8.1.44 UninstallIcon

[path\]icon.ico

Sets the icon of the uninstaller.

4.8.1.45 UninstallSubCaption

page_number subcaption

Sets the default subcaptions for the uninstaller pages (0=": Confirmation",1=": Uninstalling Files",2=": Completed"). If you specify an empty string (""), the default will be used (you can specify " " to simulate a empty string).

You can also set a subcaption (or override the default) using Caption inside a PageEx block.

Accepts variables. If variables are used, they must be initialized before the relevant page is created.

4.8.1.46 UninstallText

text [subtext]

Specifies the texts on the uninstaller confirm page.

text: Text above the controls

subtext: Text next to the uninstall location

Accepts variables. If variables are used, they must be initialized before the uninstaller confirm page is created.

4.8.1.47 Windowlcon

on | off

Sets whether or not the installer's icon is displayed on certain pages.

4.8.1.48 XPStyle

on | off

Sets whether or not a XP visual style manifest will be added to the installer. This manifest makes the installers controls use the new visual

styles when running on Windows XP and later. This affects the uninstaller too.

4.8.2 Compiler Flags

The following commands affect how the compiler generates code and compresses data. Unless otherwise noted, these commands are valid anywhere in the script and affect every line below where each one is placed (until overridden by another command). They cannot be jumped over using flow control instructions.

For example, in the following script, blah.dat will never be overwritten.

```
${If} $0 == 0
   SetOverwrite on
${Else}
   SetOverwrite off
${EndIf}
File blah.dat # overwrite is always off here!
```

Instead, the following should be used.

```
${If} $0 == 0
   SetOverwrite on
   File blah.dat
${Else}
   SetOverwrite off
   File blah.dat
${EndIf}
```

4.8.2.1 AllowSkipFiles

on off

This command specifies whether the user should be able to skip a file or not. A user has an option to skip a file if SetOverwrite is set to on (default) and the installer fails to open a file for writing when trying to extract a file.

If *off* is used the ignore button which allows the user to skip the file will not be shown and the user will only have an option to abort the installation (Cancel button) or retry opening the file for writing (Retry button). If *on* is used the user will have an option to skip the file (error flag will be set - see SetOverwrite).

4.8.2.2 FileBufSize

buffer size in mb

This command sets the size of the compiler's internal file buffers. This command allows you to control the compiler's memory usage by limiting how much of a given file it will load into memory at once. Since the compiler needs both input and output, twice the memory size specified could be used at any given time for file buffers. This command does not limit the compression buffers which could take another couple of MB, neither does it limit the compiler's other internal buffers, but those shouldn't normally top 1MB anyway. Specifying a very small number could decrease performance. Specifying a very large number could exhaust system resources and force the compiler to cancel the compilation process. The default value is 32MB.

4.8.2.3 SetCompress

auto | force | off

This command sets the compress flag which is used by the installer to determine whether or not data should be compressed. Typically the SetCompress flag will affect the commands after it, and the last SetCompress command in the file also determines whether or not the install info section and uninstall data of the installer is compressed. If compressflag is 'auto', then files are compressed if the compressed size is smaller than the uncompressed size. If compressflag is set to 'force', then the compressed version is always used. If compressflag is 'off' then compression is not used (which can be faster).

Note that this option has no effect when solid compression is used.

4.8.2.4 SetCompressor

[/SOLID] [/FINAL] **zlib**|bzip2|lzma

This command sets the compression algorithm used to compress files/data in the installer. It can only be used outside of sections and functions and before any data is compressed. Different compression methods can not be used for different files in the same installer. It is recommended to use it at the very top of the script to avoid compilation errors.

Three compression methods are supported: ZLIB, BZIP2 and LZMA.

ZLIB (the default) uses the deflate algorithm, it is a quick and simple method. With the default compression level it uses about 300 KB of memory.

BZIP2 usually gives better compression ratios than ZLIB, but it is a bit slower and uses more memory. With the default compression level it uses about 4 MB of memory.

LZMA is a new compression method that gives very good compression ratios. The decompression speed is high (10-20 MB/s on a 2 GHz CPU), the compression speed is lower. The memory size that will be used for decompression is the dictionary size plus a few KBs, the default is 8 MB.

If /FINAL is used, subsequent calls to SetCompressor will be ignored.

If /SOLID is used, all of the installer data is compressed in one block. This results in greater compression ratios.

4.8.2.5 SetCompressorDictSize

dict_size_mb

Sets the dictionary size in megabytes (MB) used by the LZMA compressor (default is 8 MB).

4.8.2.6 SetDatablockOptimize

on off

This command tells the compiler whether or not to do datablock optimizations. Datablock optimizations causes the compiler to check to see if any data being added to the data block is already in the data block, and if so, it is simply referenced as opposed to added (can save a little bit of size). It is highly recommended to leave this option on.

4.8.2.7 SetDateSave

on off

This command sets the file date/time saving flag which is used by the File command to determine whether or not to save the last write date and time of the file, so that it can be restored on installation. Valid flags are 'on' and 'off'. 'on' is the default.

4.8.2.8 SetOverwrite

on|off|try|ifnewer|ifdiff|lastused

This command sets the overwrite flag which is used by the File command to determine whether or not the file should overwrite any existing files that are present. If overwriteflag is 'on', files are overwritten (this is the default). If overwriteflag is 'off', files that are already present are not overwritten. If overwriteflag is 'try', files are overwritten if possible (meaning that if the file is not able to be written to, it is skipped without any user interaction). If overwriteflag is 'ifnewer', then files are only overwritten if the existing file is older than the new file. If overwriteflag is 'ifdiff', then files are only overwritten if the existing file is older or newer

than the new file. Note that when in 'ifnewer' or 'ifdiff' mode, the destination file's date is set, regardless of what SetDateSave is set to.

SetOverwrite off
File program.cfg # config file we don't want to overwr
SetOverwrite on

4.8.2.9 Unicode

true| false

Generate a Unicode installer. It can only be used outside of sections and functions and before any data is compressed.

4.8.3 Version Information

4.8.3.1 VIAddVersionKey

[/LANG=lang id] keyname value

Adds a string entry to the version information stored in the installer and uninstaller. These can be viewed in the File Properties Version or Details tab. keyname can either be a special name known by Windows or a user defined name. /LANG=0 can be used to indicate a language neutral language id. The following names are known by Windows:

- ProductName
- Comments
- CompanyName
- LegalCopyright
- FileDescription
- FileVersion
- ProductVersion
- InternalName
- LegalTrademarks
- OriginalFilename

- PrivateBuild
- SpecialBuild

The displayed name of these special entries are translated on the target system, whereas user defined keynames remain untranslated.

```
VIAddVersionKey /LANG=${LANG_ENGLISH} "ProductName" "T VIAddVersionKey /LANG=${LANG_ENGLISH} "Comments" "A te VIAddVersionKey /LANG=${LANG_ENGLISH} "CompanyName" "F VIAddVersionKey /LANG=${LANG_ENGLISH} "LegalTrademarks VIAddVersionKey /LANG=${LANG_ENGLISH} "LegalCopyright" VIAddVersionKey /LANG=${LANG_ENGLISH} "FileDescription VIAddVersionKey /LANG=${LANG_ENGLISH} "FileVersion" "1
```

4.8.3.2 VIProductVersion

```
[version_string_X.X.X.X]
```

Sets the Product Version in the VS_FIXEDFILEINFO version information block.

VIProductVersion 1.2.3.4

4.8.3.3 VIFileVersion

```
[version string X.X.X.X]
```

Sets the File Version in the VS_FIXEDFILEINFO version information block (You should also set the FileVersion string with VIAddVersionKey so the information is displayed at the top of the Version Tab in the Properties of the file). If you don't provide a File Version the Product Version is used in the VS_FIXEDFILEINFO block.

VIFileVersion 1.2.3.4

Previous | Contents | Next

Previous | Contents | Next

4.9 Instructions

4.9.1 Basic Instructions

The instructions that NSIS uses for scripting are sort of a cross between PHP and assembly. There are no real high level language constructs but the instructions themselves are (for the most part) high level, and you have handy string capability (i.e. you don't have to worry about concatenating strings, etc). You essentially have 25 registers (20 general purpose, 5 special purpose), and a stack.

4.9.1.1 Delete

[/REB00T0K] file

Delete file (which can be a file or wildcard, but should be specified with a full path) from the target system. If /REBOOTOK is specified and the file cannot be deleted then the file is deleted when the system reboots -- if the file will be deleted on a reboot, the reboot flag will be set. The error flag is set if files are found and cannot be deleted. The error flag is not set when trying to delete a file that does not exist.

Delete \$INSTDIR\somefile.dat

4.9.1.2 Exec

command

Execute the specified program and continue immediately. Note that the file specified must exist on the target system, not the compiling system. \$OUTDIR is used as the working directory. The error flag is set if the process could not be launched. Note, if the command could have spaces, you should put it in quotes to delimit it from parameters. e.g.: Exec "\\$INSTDIR\command.exe" parameters'. If you don't put it in quotes it will not work on Windows 9x with or without parameters.

```
Exec '"$INSTDIR\someprogram.exe"'
Exec '"$INSTDIR\someprogram.exe" some parameters'
```

4.9.1.3 ExecShell

```
action command [parameters] [SW SHOWDEFAULT | SW SHOWN
```

Execute the specified program using ShellExecute. Note that action is usually "open", "print", etc, but can be an empty string to use the default action. Parameters and the show type are optional. \$OUTDIR is used as the working directory. The error flag is set if the process could not be launched.

```
ExecShell "open" "http://nsis.sf.net/"
ExecShell "open" "$INSTDIR\readme.txt"
ExecShell "print" "$INSTDIR\readme.txt"
```

4.9.1.4 ExecWait

```
command [user_var(exit code)]
```

Execute the specified program and wait for the executed process to quit. See Exec for more information. If no output variable is specified ExecWait sets the error flag if the program executed returns a nonzero error code, or if there is an error. If an output variable is specified, ExecWait sets the variable with the exit code (and only sets the error flag if an error occurs; if an error occurs the contents of the user variable are undefined). Note, if the command could have spaces, you should put it in quotes to delimit it from parameters. e.g.: ExecWait ""\$INSTDIR\command.exe" parameters'. If you don't put it in quotes it will *not* work on Windows 9x with or without parameters.

```
ExecWait '"$INSTDIR\someprogram.exe"'
ExecWait '"$INSTDIR\someprogram.exe"' $0
DetailPrint "some program returned $0"
```

4.9.1.5 File

[/nonfatal] [/a] ([/r] [/x file|wildcard [...]] (file|

Adds file(s) to be extracted to the current output path (\$OUTDIR).

- Note that the output file name is \$OUTDIR\filename_portion_of_file.
- Use /oname=X switch to change the output name. X may contain variables and can be a fully qualified path or a relative path in which case it will be appended to \$OUTDIR set by SetOutPath. When using this switch, only one file can be specified. If the output name contains spaces, quote the entire parameter, including /oname, as shown in the examples below.
- Wildcards are supported.
- If the /r switch is used, matching files and directories are recursively searched for in subdirectories. If just one path segment is specified (e.g. File /r something), the current directory will be recursively searched. If more than one segment is specified (e.g. File /r something*.*), the last path segment will be used as the matching condition and anything before it specifies which directory to search recursively. If a directory name matches, all of its contents is added recursively. Directory structure is preserved.
- Use the /x switch to exclude files and directories.
- If the /a switch is used, the attributes of the file(s) added will be preserved.
- The File command sets the error flag if overwrite mode is set to 'try' and the file could not be overwritten, or if the overwrite mode is set to 'on' and the file could not be overwritten and the user selects ignore.
- If the /nonfatal switch is used and no files are found, a warning will be issued instead of an error.

```
File something.exe
File /a something.exe
File *.exe
File /r *.dat
File /r data
File /oname=temp.dat somefile.ext
```

```
File /oname=$TEMP\temp.dat somefile.ext
File "/oname=$TEMP\name with spaces.dat" somefile.ext
File /nonfatal "a file that might not exist"
File /r /x CVS myproject\*.*
File /r /x *.res /x *.obj /x *.pch source\*.*
```

Note: when using the /r switch, both matching directories and files will be searched. This is always done with or without the use of wildcards, even if the given path perfectly matches one directory. That means, the following directory structure:

```
<DIR> something
  file.dat
  another.dat
<DIR> dir
  something
  <DIR> dir2
    file2.dat
<DIR> another
  <DIR> something
  readme.txt
```

with the following File usage:

File /r something

will match the directory named *something* in the root directory, the file named *something* in the directory named *dir* and the directory named *something* in the directory named *another*. To match only the directory named *something* in the root directory, use the following:

```
File /r something\*.*
```

When adding *.*, it will be used as the matching condition and *something* will be used as the directory to search. When only *something* is specified, the current directory will be recursively searched for every file and directory named *something* and *another\something* will be matched.

4.9.1.6 Rename

[/REB00T0K] source_file dest_file

Rename source_file to dest_file. You can use it to move a file from anywhere on the system to anywhere else and you can move a directory to somewhere else on the same drive. The destination file must not exist or the move will fail (unless you are using /REBOOTOK). If /REBOOTOK is specified, and the file cannot be moved (if, for example, the destination exists), then the file is moved when the system reboots. If the file will be moved on a reboot, the reboot flag will be set. The error flag is set if the file cannot be renamed (and /REBOOTOK is not used) or if the source file does not exist.

If no absolute path is specified the current folder will be used. The current folder is the folder set using the last SetOutPath instruction. If you have not used SetOutPath the current folder is \$EXEDIR.

Rename \$INSTDIR\file.ext \$INSTDIR\file.dat

4.9.1.7 ReserveFile

[/nonfatal] [/r] [/x file|wildcard [...]] file [file..

Reserves a file in the data block for later use. Files are added to the compressed data block in the order they appear in the script. Functions, however, are not necessarily called in the order they appear in the script. Therefore, if you add a file in a function called early but put the function at the end of the script, all of the files added earlier will have to be decompressed to get to the required file. This process can take a long time if there a lot of files. .onlnit is one such function. It is called at the very beginning, before anything else appears. If you put it at the very end of the script, extract some files in it and have lots of files added before it, the installer might take a very long time to load. This is where this command comes useful, allowing you to speed up the loading process by including the file at the top of the data block instead of letting NSIS seek

all the way down to the bottom of the *compressed* data block.

Use /plugin to reserve a plugin in \${NSISDIR}\Plugins*.

See File for more information about the parameters.

4.9.1.8 RMDir

[/r] [/REB00T0K] directory name

Remove the specified directory (fully qualified path with no wildcards). Without /r, the directory will only be removed if it is completely empty. If /r is specified the directory will be removed recursively, so all directories and files in the specified directory will be removed. If /REBOOTOK is specified, any file or directory which could not be removed during the process will be removed on reboot -- if any file or directory will be removed on a reboot, the reboot flag will be set. The error flag is set if any file or directory cannot be removed.

```
RMDir $INSTDIR
RMDir $INSTDIR\data
RMDir /r /REB00TOK $INSTDIR
RMDir /REB00TOK $INSTDIR\DLLs
```

Note that the current working directory can not be deleted. The current working directory is set by SetOutPath. For example, the following example will not delete the directory.

```
SetOutPath $TEMP\dir
RMDir $TEMP\dir
```

The next example will succeed in deleting the directory.

```
SetOutPath $TEMP\dir
SetOutPath $TEMP
RMDir $TEMP\dir
```

Warning: using RMDir /r \$INSTDIR in the uninstaller is not safe. Though

it is unlikely, the user might select to install to the Program Files folder and so this command will wipe out the entire Program Files folder, including other programs that has nothing to do with the uninstaller. The user can also put other files but the program's files and would expect them to get deleted with the program. Solutions are available for easily uninstalling only files which were installed by the installer.

4.9.1.9 SetOutPath

outpath

Sets the output path (\$OUTDIR) and creates it (recursively if necessary), if it does not exist. Must be a full pathname, usually is just \$INSTDIR (you can specify \$INSTDIR if you are lazy with a single "-").

SetOutPath \$INSTDIR File program.exe

4.9.2 Registry, INI, File Instructions

In all of the below registry instructions use an empty string (just two quotes with nothing between them - "") as the key name to specify the default key which is shown as (Default) in regedit.exe.

If a full path is not specified for any of the INI handling instructions, the Windows directory will be used.

4.9.2.1 DeletelNISec

ini_filename section_name

Deletes the entire section [section_name] from ini_filename. If the section could not be removed from the ini file, the error flag is set. It does not set the error flag if the section could not be found.

WriteINIStr \$TEMP\something.ini section1 something 123

WriteINIStr \$TEMP\something.ini section1 somethingelse WriteINIStr \$TEMP\something.ini section2 nsis true DeleteINISec \$TEMP\something.ini section1

4.9.2.2 DeletelNIStr

```
ini_filename section_name str_name
```

Deletes the string str_name from section [section_name] from ini_filename. If the string could not be removed from the ini file, the error flag is set. It does not set the error flag if the string could not be found.

WriteINIStr \$TEMP\something.ini section1 something 123 WriteINIStr \$TEMP\something.ini section1 somethingelse DeleteINIStr \$TEMP\something.ini section1 somethingels

4.9.2.3 DeleteRegKey

```
[/ifempty] root key subkey
```

Deletes a registry key. If /ifempty is specified, the registry key will only be deleted if it has no subkeys (otherwise, the whole registry tree will be removed). Valid values for root_key are listed under WriteRegStr. The error flag is set if the key could not be removed from the registry (or if it didn't exist to begin with).

DeleteRegKey HKLM "Software\My Company\My Software"
DeleteRegKey /ifempty HKLM "Software\A key that might

4.9.2.4 DeleteRegValue

```
root_key subkey key_name
```

Deletes a registry value. Valid values for root_key are listed under WriteRegStr. The error flag is set if the value could not be removed from the registry (or if it didn't exist to begin with).

4.9.2.5 EnumRegKey

```
user var(output) root key subkey index
```

Set user variable \$x with the name of the 'index'th registry key in root_key\Subkey. Valid values for root_key are listed under WriteRegStr. Returns an empty string if there are no more keys, and returns an empty string and sets the error flag if there is an error.

```
StrCpy $0 0
loop:
    EnumRegKey $1 HKLM Software $0
    StrCmp $1 "" done
    IntOp $0 $0 + 1
    MessageBox MB_YESNO|MB_ICONQUESTION "$1$\n$\nMore?"
done:
```

4.9.2.6 EnumRegValue

```
user_var(output) root_key subkey index
```

Set user variable \$x with the name of the 'index'th registry value in root_key\Subkey. Valid values for root_key are listed under WriteRegStr. Returns an empty string and sets the error flag if there are no more values or if there is an error.

```
StrCpy $0 0
loop:
   ClearErrors
   EnumRegValue $1 HKLM Software\Microsoft\Windows\Curr
   IfErrors done
   IntOp $0 $0 + 1
   ReadRegStr $2 HKLM Software\Microsoft\Windows\Curren
   MessageBox MB_YESNO|MB_ICONQUESTION "$1 = $2$\n$\nMo
```

done:

4.9.2.7 ExpandEnvStrings

```
user var(output) string
```

Expands environment variables in *string* into the user variable \$x. If an environment variable doesn't exist, it will not be replaced. For example, if you use "%var%" and var doesn't exists, the result will be "%var%". If there is an error, the variable is set to empty, and the error flag is set.

ExpandEnvStrings \$0 "WINDIR=%WINDIR%\$\nTEMP=%TEMP%"

4.9.2.8 FlushINI

```
ini_filename
```

Flushes the INI file's buffers. Windows 9x keeps all changes to the INI file in memory. This command causes the changes to be written to the disk immediately. Use it if you edit the INI manually, delete it, move it or copy it right after you change it with WriteINIStr, DeleteINISec or DeleteINStr.

```
WriteINIStr $TEMP\something.ini test test
FlushINI $TEMP\something.ini
Delete $TEMP\something.ini
```

4.9.2.9 ReadEnvStr

```
user_var(output) name
```

Reads from the environment string "name" and sets the value into the user variable \$x. If there is an error reading the string, the user variable is set to empty, and the error flag is set.

```
ReadEnvStr $0 WINDIR
ReadEnvStr $1 TEMP
```

4.9.2.10 ReadINIStr

user_var(output) ini_filename section_name entry_name

Reads from entry_name in [section_name] of ini_filename and stores the value into user variable \$x. The error flag will be set and \$x will be assigned to an empty string if the entry is not found.

ReadINIStr \$0 \$INSTDIR\winamp.ini winamp outname

4.9.2.11 ReadRegDWORD

```
user var(output) root key sub key name
```

Reads a 32-bit DWORD from the registry into the user variable \$x. Valid values for root_key are listed under WriteRegStr. The error flag will be set and \$x will be set to an empty string ("" which is interpreted as 0 in math operations) if the DWORD is not present. If the value is present, but is not a DWORD, it will be read as a string and the error flag will be set.

ReadRegDWORD \$0 HKLM Software\NSIS VersionBuild

4.9.2.12 ReadRegStr

```
user_var(output) root_key sub_key name
```

Reads from the registry into the user variable \$x. Valid values for root_key are listed under WriteRegStr. The error flag will be set and \$x will be set to an empty string ("") if the string is not present. If the value is present, but is of type REG_DWORD, it will be read and converted to a string and the error flag will be set.

```
ReadRegStr $0 HKLM Software\NSIS ""
DetailPrint "NSIS is installed at: $0"
```

4.9.2.13 WriteINIStr

```
ini_filename section_name entry_name value
```

Writes entry_name=value into [section_name] of ini_filename. The error flag is set if the string could not be written to the ini file.

```
WriteINIStr $TEMP\something.ini section1 something 123 WriteINIStr $TEMP\something.ini section1 somethingelse WriteINIStr $TEMP\something.ini section2 nsis true
```

4.9.2.14 WriteRegBin

```
root_key subkey key_name valuedata
```

This command writes a block of binary data to the registry. Valid values for root_key are listed under WriteRegStr. Valuedata is in hexadecimal (e.g. DEADBEEF01223211151). The error flag is set if the binary data could not be written to the registry. If the registry key doesn't exist it will be created.

WriteRegBin HKLM "Software\My Company\My Software" "Bi

4.9.2.15 WriteRegDWORD

```
root_key subkey key_name value
```

This command writes a DWORD (32-bit integer) to the registry (a user variable can be specified). Valid values for root_key are listed under WriteRegStr. The error flag is set if the dword could not be written to the registry. If the registry key doesn't exist it will be created.

WriteRegDWORD HKLM "Software\My Company\My Software" "

4.9.2.16 WriteRegStr

root_key subkey key_name value

Write a string to the registry. See WriteRegExpandStr for more details.

WriteRegStr HKLM "Software\My Company\My Software" "St

4.9.2.17 WriteRegExpandStr

root key subkey key name value

Write a string to the registry. *root_key* must be one of:

- HKCR or HKEY CLASSES ROOT
- HKLM or HKEY LOCAL MACHINE
- HKCU or HKEY CURRENT USER
- HKU or HKEY USERS
- HKCC or HKEY CURRENT CONFIG
- HKDD or HKEY_DYN_DATA
- HKPD or HKEY_PERFORMANCE_DATA
- SHCTX or SHELL CONTEXT

If root_key is SHCTX or SHELL_CONTEXT, it will be replaced with HKLM if SetShellVarContext is set to all and with HKCU if SetShellVarContext is set to current.

The error flag is set if the string could not be written to the registry. The type of the string will be REG_SZ for WriteRegStr, or REG_EXPAND_STR for WriteRegExpandStr. If the registry key doesn't exist it will be created.

WriteRegExpandStr HKLM "Software\My Company\My Softwar

4.9.3 General Purpose Instructions

4.9.3.1 CallInstDLL

dllfile function name

Calls a function named function name inside a NSIS extension DLL, a

plug-in. See the example plugin for how to make one. Extension DLLs can access the stack and variables. Note: To automatically extract and call plug-in DLLs, use a plug-in command instead of CallInstDLL.

```
Push "a parameter"
Push "another parameter"
CallInstDLL $INSTDIR\somedll.dll somefunction
```

For easier plug-in handling, use the new plug-in call syntax.

4.9.3.2 CopyFiles

```
[/SILENT] [/FILESONLY] filespec on destsys destination
```

Copies files from the source to the destination on the installing system. Useful with \$EXEDIR if you want to copy from installation media, or to copy from one place to another on the system. You might see a Windows status window of the copy operation if the operation takes a lot of time (to disable this, use /SILENT). The last parameter can be used to specify the size of the files that will be copied (in kilobytes), so that the installer can approximate the disk space requirements. On error, or if the user cancels the copy (only possible when /SILENT was omitted), the error flag is set. If /FILESONLY is specified, only files are copied.

Fully-qualified path names should always be used with this instruction. Using relative paths will have unpredictable results.

```
CreateDirectory $INSTDIR\backup
CopyFiles $INSTDIR\*.dat $INSTDIR\backup
```

4.9.3.3 CreateDirectory

```
path_to_create
```

Creates (recursively if necessary) the specified directory. The error flag is set if the directory couldn't be created.

You should always specify an absolute path.

CreateDirectory \$INSTDIR\some\directory

4.9.3.4 CreateShortcut

[/NoWorkingDir] link.lnk target.file [parameters [icon

Creates a shortcut 'link.lnk' that links to 'target.file', with optional parameters 'parameters'. The icon used for the shortcut is 'icon.file,icon_index_number'; for default icon settings use empty strings for both icon.file and icon_index_number. start_options should be one of: \$\$SW_SHOWNORMAL, \$\$SW_SHOWMAXIMIZED, \$\$SW_SHOWMINIMIZED, or an empty string. keyboard_shortcut should be in the form of 'flag|c' where flag can be a combination (using |) of: \$\$ALT, \$CONTROL, \$EXT, or \$\$SHIFT. c is the character to use (a-z, A-Z, 0-9, F1-F24, etc). Note that no spaces are allowed in this string. A good example is "ALT|CONTROL|F8". \$OUTDIR is used as the working directory. You can change it by using \$\$SetOutPath before creating the Shortcut or use /NoWorkingDir if you don't need to set the working directory. description should be the description of the shortcut, or comment as it is called under XP. The error flag is set if the shortcut cannot be created (i.e. either of the paths (link or target) does not exist, or some other error).

CreateDirectory "\$SMPROGRAMS\My Company"
CreateShortcut "\$SMPROGRAMS\My Company\My Program.lnk"
 "some command line parameters" "\$INSTDIR\My Program.
 ALT|CONTROL|SHIFT|F5 "a description"

4.9.3.5 GetDLLVersion

filename user var(high dword output) user var(low dwor

Gets the version information from the DLL (or any other executable containing version information) in "filename". Sets the user output variables with the high and low dwords of version information on success;

on failure the outputs are empty and the error flag is set. The following example reads the DLL version and copies a human readable version of it into \$0:

```
GetDllVersion "$INSTDIR\MyDLL.dll" $R0 $R1
IntOp $R2 $R0 / 0x00010000
IntOp $R3 $R0 & 0x0000FFFF
IntOp $R4 $R1 / 0x00010000
IntOp $R5 $R1 & 0x0000FFFF
StrCpy $0 "$R2.$R3.$R4.$R5"
```

4.9.3.6 GetDLLVersionLocal

localfilename user_var(high dword output) user_var(low

This is similar to GetDLLVersion, only it acts on the system building the installer (it actually compiles into two StrCpy commands). Sets the two output variables with the DLL version information of the DLL on the build system. Use !getdllversion if you need to use the values with VIProductVersion.

4.9.3.7 GetFileTime

filename user_var(high dword output) user_var(low dwor

Gets the last write time of "filename". Sets the user output variables with the high and low dwords of the FILETIME timestamp on success; on failure the outputs are empty and the error flag is set.

4.9.3.8 GetFileTimeLocal

localfilename user var(high dword output) user var(low

This is similar to GetFileTime, only it acts on the system building the installer (it actually compiles into two StrCpy commands). Sets the two

output variables with the file timestamp of the file on the build system.

4.9.3.9 GetFullPathName

```
[/SHORT] user_var(output) path_or_file
```

Assign the full path of the file specified to user variable \$x. If the path portion of the parameter is not found, the error flag will be set and \$x will be empty. If /SHORT is specified, the path is converted to the short filename form. However, if /SHORT is not specified, the path isn't converted to its long filename form. To get the long filename, call GetLongPathName using the System plug-in. Note that GetLongPathName is only available on Windows 98, Windows 2000 and above.

```
StrCpy $INSTDIR $PROGRAMFILES\NSIS
SetOutPath $INSTDIR
GetFullPathName $0 ..
DetailPrint $0 # will print C:\Program Files
GetFullPathName /SHORT $0 $INSTDIR
DetailPrint $0 # will print C:\Progra~1\NSIS
```

```
StrCpy $0 C:\Progra~1\NSIS
System::Call 'kernel32::GetLongPathName(t r0, t .r1, i
StrCmp $2 error +2
StrCpy $0 $1
DetailPrint $0 # will print C:\Program Files\NSIS, whe
```

4.9.3.10 GetTempFileName

```
user_var(output) [base_dir]
```

Assign to the user variable \$x, the name of a temporary file. The file will be created for you and it will be empty. The name of the temporary file is guaranteed to be unique. If to want the temporary file to be created in another directory other than the Windows temp directory, specify a

base_dir. You should Delete the file when you are done with it.

```
GetTempFileName $0
File /oname=$0 something.dat
# do something with something.dat
Delete $0
```

4.9.3.11 SearchPath

```
user var(output) filename
```

Assign to the user variable \$x, the full path of the file named by the second parameter. The error flag will be set and \$x will be empty if the file cannot be found. Uses SearchPath() to search the system paths for the file.

4.9.3.12 SetFileAttributes

```
filename attribute1|attribute2|...
```

Sets the file attributes of 'filename'. Valid attributes can be combined with and are:

- NORMAL or FILE_ATTRIBUTE_NORMAL (you can use 0 to abbreviate this)
- ARCHIVE or FILE ATTRIBUTE ARCHIVE
- HIDDEN or FILE_ATTRIBUTE_HIDDEN
- OFFLINE or FILE ATTRIBUTE OFFLINE
- READONLY or FILE_ATTRIBUTE_READONLY
- SYSTEM or FILE_ATTRIBUTE_SYSTEM
- TEMPORARY or FILE_ATTRIBUTE_TEMPORARY

The error flag will be set if the file's attributes cannot be set (i.e. the file doesn't exist, or you don't have the right permissions). You can only set attributes. It's not possible to unset them. If you want to remove an attribute use NORMAL. This way all attributes are erased. This command

doesn't support wildcards.

4.9.3.13 RegDLL

dllfile [entrypoint_name]

Loads the specified DLL and calls DllRegisterServer (or entrypoint_name if specified). The error flag is set if an error occurs (i.e. it can't load the DLL, initialize OLE, find the entry point, or the function returned anything other than ERROR_SUCCESS (=0)).

Use SetOutPath to set the current directory for DLLs that depend on other DLLs that are now in the path or in the Windows directory. For example, if foo.dll depends on bar.dll which is located in \$INSTDIR use:

SetOutPath \$INSTDIR
RegDLL \$INSTDIR\foo.dll

4.9.3.14 UnRegDLL

dllfile

Loads the specified DLL and calls DllUnregisterServer. The error flag is set if an error occurs (i.e. it can't load the DLL, initialize OLE, find the entry point, or the function returned anything other than ERROR SUCCESS (=0)).

4.9.4 Flow Control Instructions

4.9.4.1 Abort

[user_message]

Cancels the install, stops execution of script, and displays user_message in the status display. Note: you can use this from Callback functions to do special things. Page callbacks also uses Abort for special purposes.

```
Abort
Abort "can't install"
```

4.9.4.2 Call

```
function_name | :label_name | user_var(input)
```

Calls the function named *function_name*, the label named *label_name*, or a variable that specifies an address. An address is returned by GetCurrentAddress, GetFunctionAddress or GetLabelAddress. A call returns when it encounters a Return instruction. Sections and functions are automatically ended with a Return instruction. Uninstall functions cannot be called from installer functions and sections, and vice-versa.

```
Function func
   Call :label
   DetailPrint "#1: This will only appear 1 time."
label:
   DetailPrint "#2: This will appear before and after m
   Call :.global_label
FunctionEnd

Section
   Call func
   Return

.global_label:
   DetailPrint "#3: The global label was called"
SectionEnd
```

4.9.4.3 ClearErrors

Clears the error flag.

```
ClearErrors
IfErrors 0 +2
```

4.9.4.4 GetCurrentAddress

```
user_var(output)
```

Gets the address of the current instruction (the GetCurrentAddress) and stores it in the output user variable. This user variable then can be passed to Call or Goto.

```
Function func
  DetailPrint "function"
  IntOp $0 $0 + 2 ; Calculate the address after of the
  Call $0
  DetailPrint "function end"
FunctionEnd
Section
 DetailPrint "section"
  GetCurrentAddress $0
  Goto callFunc
  DetailPrint "back in section"
  Return
callFunc:
  Call func
  DetailPrint "section end"
SectionEnd
```

4.9.4.5 GetFunctionAddress

```
user_var(output) function_name
```

Gets the address of the function and stores it in the output user variable. This user variable then can be passed to Call or Goto. Note that if you

Goto an address which is the output of GetFunctionAddress, your function will never be returned to (when the function you Goto'd to returns, you return instantly).

```
Function func
DetailPrint "function"
FunctionEnd

Section
GetFunctionAddress $0 func
Call $0
SectionEnd
```

4.9.4.6 GetLabelAddress

```
user_var(output) label
```

Gets the address of the label and stores it in the output user variable. This user variable then can be passed to Call or Goto. Note that you may only call this with labels accessible from your function, but you can call it from anywhere (which is potentially dangerous). Note that if you Call the output of GetLabelAddress, code will be executed until it Return's (explicitly or implicitly at the end of a function), and then you will be returned to the statement after the Call.

```
label:
DetailPrint "label"
GetLabelAddress $0 label
IntOp $0 $0 + 4
Goto $0
DetailPrint "done"
```

4.9.4.7 Goto

```
label_to_jump_to | +offset| -offset| user_var(target)
```

If label is specified, goto the label 'label_to_jump_to:'.

If +offset or -offset is specified, jump is relative by offset instructions. Goto +1 goes to the next instruction, Goto -1 goes to the previous instruction, etc.

If a user variable is specified, jumps to absolute address (generally you will want to get this value from a function like GetLabelAddress). Compiler flag commands and SectionIn aren't instructions so jumping over them has no effect.

```
Goto label
Goto +2
Goto -2
Goto $0
```

4.9.4.8 IfAbort

```
label_to_goto_if_abort [label_to_goto_if_no_abort]
```

Will "return" true if the installation has been aborted. This can happen if the user chose abort on a file that failed to create (or overwrite) or if the user aborted by hand. This function can only be called from the leave function of the instfiles page.

```
Page instfiles "" "" instfilesLeave

Function instfilesLeave
   IfAbort 0 +2
    MessageBox MB_OK "user aborted"

FunctionEnd
```

4.9.4.9 IfErrors

```
jumpto_iferror [jumpto_ifnoerror]
```

Checks and clears the error flag, and if it is set, it will goto jumpto_iferror, otherwise it will goto jumpto_ifnoerror. The error flag is set by other instructions when a recoverable error (such as trying to delete a file that is in use) occurs.

```
ClearErrors
File file.dat
IfErrors 0 +2
  Call ErrorHandler
```

4.9.4.10 IfFileExists

```
file to check for jump if present [jump otherwise]
```

Checks for existence of file(s) file_to_check_for (which can be a wildcard, or a directory), and Gotos jump_if_present if the file exists, otherwise Gotos jump_otherwise. If you want to check to see if a file is a directory, use IfFileExists DIRECTORY*.*

```
IfFileExists $WINDIR\notepad.exe 0 +2
   MessageBox MB_OK "notepad is installed"
```

4.9.4.11 IfRebootFlag

```
jump_if_set [jump_if_not_set]
```

Checks the reboot flag, and jumps to jump_if_set if the reboot flag is set, otherwise jumps to jump_if_not_set. The reboot flag can be set by Delete and Rename, or manually with SetRebootFlag.

```
IfRebootFlag 0 noreboot
   MessageBox MB_YESNO "A reboot is required to finish
   Reboot
noreboot:
```

4.9.4.12 IfSilent

```
jump_if_silent [jump_if_not]
```

Checks the silent flag, and jumps to jump_if_silent if the installer is silent, otherwise jumps to jump_if_not. The silent flag can be set by SilentInstall, SilentUninstall, SetSilent and by the user passing /S on the command line.

```
IfSilent +2
   ExecWait '"$INSTDIR\nonsilentprogram.exe"'
```

4.9.4.13 IntCmp

```
val1 val2 jump_if_equal [jump_if_val1_less] [jump_if_v
```

Compares two integers val1 and val2. If val1 and val2 are equal, Gotos jump_if_equal, otherwise if val1 < val2, Gotos jump_if_val1_less, otherwise if val1 > val2, Gotos jump_if_val1_more.

```
IntCmp $0 5 is5 lessthan5 morethan5
is5:
   DetailPrint "$$0 == 5"
   Goto done
lessthan5:
   DetailPrint "$$0 < 5"
   Goto done
morethan5:
   DetailPrint "$$0 > 5"
   Goto done
done:
```

4.9.4.14 IntCmpU

```
val1 val2 jump_if_equal [jump_if_val1_less] [jump_if_v Same as IntCmp, but treats the values as unsigned integers.
```

4.9.4.15 MessageBox

mb_option_list messagebox_text [/SD return] [return_ch

Displays a MessageBox containing the text "messagebox_text". mb_option_list must be one or more of the following, delimited by |s (e.g. MB_YESNO|MB_ICONSTOP).

- MB_OK Display with an OK button
- MB_OKCANCEL Display with an OK and a cancel button
- MB_ABORTRETRYIGNORE Display with abort, retry, ignore buttons
- MB_RETRYCANCEL Display with retry and cancel buttons
- MB_YESNO Display with yes and no buttons
- MB_YESNOCANCEL Display with yes, no, cancel buttons
- MB_ICONEXCLAMATION Display with exclamation icon
- MB_ICONINFORMATION Display with information icon
- MB_ICONQUESTION Display with question mark icon
- MB_ICONSTOP Display with stop icon
- MB_USERICON Display with installer's icon
- MB TOPMOST Make messagebox topmost
- MB_SETFOREGROUND Set foreground
- MB_RIGHT Right align text
- MB_RTLREADING RTL reading order
- MB_DEFBUTTON1 Button 1 is default
- MB DEFBUTTON2 Button 2 is default
- MB_DEFBUTTON3 Button 3 is default
- MB DEFBUTTON4 Button 4 is default

Return_check can be 0 (or empty, or left off), or one of the following:

- IDABORT Abort button
- IDCANCEL Cancel button
- IDIGNORE Ignore button
- IDNO No button
- IDOK OK button
- IDRETRY Retry button

IDYES - Yes button

If the return value of the MessageBox is return_check, the installer will Goto jumpto.

Use the /SD parameter with one of the return_check values above to specify the option that will be used when the installer is silent. See section 4.12 for more information.

```
MessageBox MB_OK "simple message box"
MessageBox MB_YESNO "is it true?" IDYES true IDNO fals
true:
    DetailPrint "it's true!"
    Goto next
false:
    DetailPrint "it's false"
next:
MessageBox MB_YESNO "is it true? (defaults to yes on s
    DetailPrint "it's true (or silent)!"
    Goto next2
false2:
    DetailPrint "it's false"
next2:
```

4.9.4.16 Return

Returns from a function or section.

```
Function func
StrCmp $0 "return now" 0 +2
Return
# do stuff
FunctionEnd

Section
Call func
; "Return" will return here
```

4.9.4.17 Quit

Causes the installer to exit as soon as possible. After Quit is called, the installer will exit (no callback functions will get a chance to run).

4.9.4.18 SetErrors

Sets the error flag.

```
SetErrors
IfErrors 0 +2
   MessageBox MB_OK "this message box will always show"
```

4.9.4.19 StrCmp

```
str1 str2 jump if equal [jump if not equal]
```

Compares (case insensitively) str1 to str2. If str1 and str2 are equal, Gotos jump_if_equal, otherwise Gotos jump_if_not_equal.

```
StrCmp $0 "a string" 0 +3
  DetailPrint '$$0 == "a string"'
  Goto +2
  DetailPrint '$$0 != "a string"'
```

4.9.4.20 StrCmpS

```
str1 str2 jump_if_equal [jump_if_not_equal]
```

Same as StrCmp, but case sensitive.

4.9.5 File Instructions

4.9.5.1 FileClose

handle

Closes a file handle opened with FileOpen.

4.9.5.2 FileOpen

user var(handle output) filename openmode

Opens a file named "filename" and sets the handle output variable with the handle. The openmode should be one of "r" (read) "w" (write, all contents of file are destroyed) or "a" (append, meaning opened for both read and write, contents preserved). In all open modes, the file pointer is placed at the beginning of the file. If the file cannot be opened the handle output is set to empty and the error flag is set.

If no absolute path is specified the current folder will be used. The current folder is the folder set using the last SetOutPath instruction. If you have not used SetOutPath the current folder is \$EXEDIR.

FileOpen \$0 \$INSTDIR\file.dat r
FileClose \$0

4.9.5.3 FileRead

handle user var(output) [maxlen]

Reads a string (ANSI characters) from a file opened with FileOpen. The string is read until either a newline (or carriage return newline pair) occurs, or until a null byte is read, or until maxlen is met (if specified). By default, strings are limited to 1024 characters (a special build with larger NSIS_MAX_STRLEN can be compiled or downloaded). If the end of file is reached and no more data is available, the output string will be empty and the error flag will be set.

Unicode: DBCS text is supported but conversion output is limited to UCS-2/BMP, surrogate pairs are not supported. The system default ANSI codepage (ACP) is used during the conversion.

```
ClearErrors
FileOpen $0 $INSTDIR\file.dat r
IfErrors done
FileRead $0 $1
DetailPrint $1
FileClose $0
done:
```

4.9.5.4 FileReadUTF16LE

```
handle user_var(output) [maxlen]
```

This function is only available when building a Unicode installer.

Reads a string (UTF-16LE characters) from a file opened with FileOpen. The string is read until either a newline (or carriage return newline pair) occurs, or until a null wide-character is read, or until maxlen is met (if specified). By default, strings are limited to 1024 characters (a special build with larger NSIS_MAX_STRLEN can be compiled or downloaded). If the end of file is reached and no more data is available, the output string will be empty and the error flag will be set. If present, the BOM at the start of the file is skipped.

```
ClearErrors
FileOpen $0 $INSTDIR\file.dat r
IfErrors done
FileReadUTF16LE $0 $1
DetailPrint $1
FileClose $0
done:
```

4.9.5.5 FileReadByte

handle user_var(output)

Reads a byte from a file opened with FileOpen. The byte is stored in the output as an integer (0-255). If the end of file is reached and no more data is available, the output will be empty and the error flag will be set.

```
ClearErrors
FileOpen $0 $INSTDIR\file.dat r
IfErrors done
FileReadByte $0 $1
FileReadByte $0 $2
DetailPrint "$1 $2"
FileClose $0
done:
```

4.9.5.6 FileReadWord

```
handle user_var(output)
```

This function is only available when building a Unicode installer.

Reads a word (2-bytes) from a file opened with FileOpen. The word is stored in the output as an integer (0-65535). If the end of file is reached and no more data is available, the output will be empty and the error flag will be set.

```
ClearErrors
FileOpen $0 $INSTDIR\file.dat r
IfErrors done
FileReadWord $0 $1
FileReadWord $0 $2
DetailPrint "$1 $2"
FileClose $0
done:
```

4.9.5.7 FileSeek

```
handle offset [mode] [user_var(new position)]
```

Seeks a file opened with FileOpen. If mode is omitted or specified as SET, the file is positioned to "offset", relative to the beginning of the file. If mode is specified as CUR, then the file is positioned to "offset", relative to the current file position. If mode is specified as END, then the file is positioned to "offset", relative to the end of the file. If the final parameter "new position" is specified, the new file position will be stored in that variable.

```
ClearErrors
FileOpen $0 $INSTDIR\file.dat r
IfErrors done
FileSeek $0 -5 END
FileRead $0 $1
DetailPrint $1
FileClose $0
done:
```

4.9.5.8 FileWrite

handle string

Writes an ANSI string to a file opened with FileOpen. If an error occurs writing, the error flag will be set.

(If you are building a Unicode installer, the function converts the string to ANSI/MBCS. The system default ANSI codepage (ACP) is used during the conversion)

```
ClearErrors
FileOpen $0 $INSTDIR\file.dat w
IfErrors done
FileWrite $0 "some text"
FileClose $0
done:
```

4.9.5.9 FileWriteUTF16LE

[/BOM] handle string

This function is only available when building a Unicode installer.

Writes a Unicode (UTF-16LE) string to a file opened with FileOpen. If an error occurs, the error flag will be set. A BOM can be added to empty files with /BOM.

```
ClearErrors
FileOpen $0 $INSTDIR\file.dat w
IfErrors done
FileWriteUTF16LE $0 "some text"
FileClose $0
done:
```

4.9.5.10 FileWriteByte

handle string

Writes the integer interpretation of 'string' to a file opened with FileOpen. The error flag is set if an error occurs while writing. The following code writes a "Carriage Return / Line Feed" pair to the file.

```
FileWriteByte file_handle "13"
FileWriteByte file_handle "10"
```

Note that only the low byte of the integer is used, i.e. writing 256 is the same as writing 0, etc.

4.9.5.11 FileWriteWord

handle string

This function is only available when building a Unicode installer.

Writes the integer interpretation of 'string' as a WORD (2-bytes, range: 0-65535) to a file opened with FileOpen. The error flag is set if an error occurs while writing. The following code writes a "Carriage Return / Line Feed" pair to the file.

```
FileWriteWord file_handle "13"
FileWriteWord file_handle "10"
```

Note that only the low WORD of the integer is used, i.e. writing 65536 is the same as writing 0, etc.

4.9.5.12 FindClose

handle

Closes a search opened with FindFirst.

4.9.5.13 FindFirst

user_var(handle output) user_var(filename output) file

Performs a search for 'filespec', placing the first file found in filename_output (a user variable). It also puts the handle of the search into handle_output (also a user variable). If no files are found, both outputs are set to empty and the error flag is set. FindClose must be used to close the handle. Note that the filename output is without path.

```
FindFirst $0 $1 $INSTDIR\*.txt
loop:
   StrCmp $1 "" done
   DetailPrint $1
   FindNext $0 $1
   Goto loop
done:
FindClose $0
```

4.9.5.14 FindNext

handle user_var(filename_output)

Continues a search began with FindFirst. handle should be the handle_output_variable returned by FindFirst. If the search is completed (there are no more files), filename_output is set to empty and the error flag is set. Note that the filename output is without path.

4.9.6 Uninstaller Instructions

4.9.6.1 WriteUninstaller

[Path\]exename.exe

Writes the uninstaller to the filename (and optionally path) specified. Only valid from within an install section or function and requires that you have an uninstall section in your script. You can call this one or more times to write out one or more copies of the uninstaller.

WriteUninstaller \$INSTDIR\uninstaller.exe

4.9.7 Miscellaneous Instructions

4.9.7.1 GetErrorLevel

user var(error level output)

Returns the last error level set by SetErrorLevel or -1 if it has never been set.

GetErrorLevel \$0
IntOp \$0 \$0 + 1
SetErrorLevel \$0

4.9.7.2 GetInstDirError

```
user_var(error output)
```

Use in the leave function of a directory page. Reads the flag set if 'DirVerify leave' is used. Possible values:

0: No error

1: Invalid installation directory

2: Not enough space on installation drive

```
!include LogicLib.nsh
PageEx directory
  DirVerify leave
  PageCallbacks "" "" dirLeave
PageExEnd
Function dirLeave
  GetInstDirError $0
  ${Switch} $0
    ${Case} 0
      MessageBox MB OK "valid installation directory"
      ${Break}
    ${Case} 1
      MessageBox MB OK "invalid installation directory
      Abort
      ${Break}
    ${Case} 2
      MessageBox MB_OK "not enough free space!"
      Abort
      ${Break}
  ${EndSwitch}
FunctionEnd
```

4.9.7.3 InitPluginsDir

Initializes the plug-ins dir (\$PLUGINSDIR) if not already initialized.

InitPluginsDir
File /oname=\$PLUGINSDIR\image.bmp image.bmp

4.9.7.4 Nop

Does nothing.

4.9.7.5 SetErrorLevel

error_level

Sets the error level of the installer or uninstaller to *error_level*. See Error Levels for more information.

IfRebootFlag 0 +2
 SetErrorLevel 4

4.9.7.6 SetRegView

32|64|lastused

Sets the registry view affected by registry commands. On 64-bit versions of Windows there are two views; one for 32-bit applications and one for 64-bit applications. By default, 32-bit applications running on 64-bit systems (WOW64) only have access to the 32-bit view. Using SetRegView 64 allows the installer to access keys in the 64-bit view of the registry.

Affects DeleteRegKey, DeleteRegValue, EnumRegKey, EnumRegValue, ReadRegDWORD, ReadRegStr, WriteRegBin, WriteRegDWORD, WriteRegExpandStr.

Does not affect InstallDirRegKey. Instead, the registry must be read using ReadRegStr in .onInit.

```
SetRegView 32
ReadRegStr $0 HKLM Software\Microsoft\Windows\CurrentV
DetailPrint $0 # prints C:\Program Files (x86)
SetRegView 64
ReadRegStr $0 HKLM Software\Microsoft\Windows\CurrentV
DetailPrint $0 # prints C:\Program Files
```

```
Function .onInit
SetRegView 64
ReadRegStr $INSTDIR HKLM Software\NSIS ""
SetRegView 32
FunctionEnd
```

4.9.7.7 SetShellVarContext

current|all

Sets the context of \$SMPROGRAMS and other shell folders. If set to 'current' (the default), the current user's shell folders are used. If set to 'all', the 'all users' shell folder is used. The all users folder may not be supported on all OSes. If the all users folder is not found, the current user folder will be used. Please take into consideration that a "normal user" has no rights to write in the all users area. Only admins have full access rights to the all users area. You can check this by using the UserInfo plug-in. See Contrib\UserInfo\UserInfo\UserInfo.nsi for an example.

Note that, if used in installer code, this will only affect the installer, and if used in uninstaller code, this will only affect the uninstaller. To affect both, it needs to be used in both.

SetShellVarContext current StrCpy \$0 \$DESKTOP SetShellVarContext all StrCpy \$1 \$DESKTOP MessageBox MB OK \$0\$\n\$1

4.9.7.8 Sleep

```
sleeptime_in_ms
```

Pauses execution in the installer for sleeptime_in_ms milliseconds. sleeptime_in_ms can be a variable, e.g. "\$0" or a number, i.e. "666".

```
DetailPrint "sleeping..."
Sleep 3000
DetailPrint "back to work"
```

4.9.8 String Manipulation Instructions

4.9.8.1 StrCpy

```
user var(destination) str [maxlen] [start offset]
```

Sets the user variable \$x with str. str can contain variables (including the user variable being set (concatenating strings this way is possible, etc)). If maxlen is specified, the string will be a maximum of maxlen characters (if maxlen is negative, the string will be truncated abs(maxlen) characters from the end). If start_offset is specified, the source is offset by it (if start_offset is negative, it will start abs(start_offset) from the end of the string).

```
StrCpy $0 "a string" # = "a string"
StrCpy $0 "a string" 3 # = "a s"
StrCpy $0 "a string" -1 # = "a strin"
StrCpy $0 "a string" "" 2 # = "string"
StrCpy $0 "a string" "" -3 # = "ing"
StrCpy $0 "a string" 3 -4 # = "rin"
StrCpy $0 "$0$0" # = "rinrin"
```

4.9.8.2 StrLen

```
user_var(length output) str
```

Sets user variable \$x to the length of str.

```
StrLen $0 "123456" # = 6
```

4.9.9 Stack Support

4.9.9.1 Exch

```
[user_var | stack_index]
```

When no parameter is specified, exchanges the top two elements of the stack. When a parameter is specified and is a user variable, exchanges the top element of the stack with the parameter. When a parameter is specified and is a positive integer, Exch will swap the item on the top of the stack with the item that is specified by the offset from the top of the stack in the parameter. If there are not enough items on the stack to accomplish the exchange, a fatal error will occur (to help you debug your code:).

```
Push 1
Push 2
Exch
Pop $0 # = 1

Push 1
Push 2
Push 3
Exch 2
Pop $0 # = 1

StrCpy $0 1
Push 2
Exch $0 # = 2
```

4.9.9.2 Pop

Pop \$1 # = 1

```
user_var(out)
```

Pops a string off of the stack into user variable \$x. If the stack is empty, the error flag will be set.

```
Push 1
Pop $0 # = 1
```

4.9.9.3 Push

```
string
```

Pushes a string onto the stack. The string can then be Pop'ed off of the stack.

Push "a string"

4.9.10 Integer Support

4.9.10.1 IntFmt

```
user var(output) format numberstring
```

Formats the number in "numberstring" using the format "format", and sets the output to user variable \$x. Example format strings include "%08X" "%u"

```
IntFmt $0 "0x%08X" 195948557
IntFmt $0 "%c" 0x41
```

4.9.10.2 IntOp

```
user var(output) value1 OP [value2]
```

Combines value1 and (depending on OP) value2 into the specified user variable (user_var). OP is defined as one of the following:

- + ADDs value1 and value2
- - SUBTRACTs value2 from value1
- * MULTIPLIEs value1 and value2
- / DIVIDEs value1 by value2
- % MODULUSs value1 by value2
- | BINARY ORs value1 and value2
- & BINARY ANDs value1 and value2
- ^ BINARY XORs value1 and value2
- >> RIGHT SHIFTs value1 by value2
- << LEFT SHIFTs value1 by value2
- ~ BITWISE NEGATEs value1 (i.e. 7 becomes 4294967288)
- ! LOGICALLY NEGATEs value1 (i.e. 7 becomes 0)
- || LOGICALLY ORs value1 and value2
- && LOGICALLY ANDs value1 and value2

```
IntOp $0 1 + 1
IntOp $0 $0 + 1
IntOp $0 $0 << 2
IntOp $0 $0 ~
IntOp $0 $0 & 0xF
```

4.9.11 Reboot Instructions

4.9.11.1 Reboot

Reboots the computer. Be careful with this one. If it fails, .onRebootFailed is called. In any case, this instruction never returns, just like Quit.

MessageBox MB_YESNO|MB_ICONQUESTION "Do you wish to re Reboot

4.9.11.2 SetRebootFlag

```
true|false
```

Sets the reboot flag to either true or false. The flag's value can be read

using IfRebootFlag.

```
SetRebootFlag true
IfRebootFlag 0 +2
   MessageBox MB_OK "this message box will always show"
```

4.9.12 Install Logging Instructions

4.9.12.1 LogSet

on | off

Sets whether install logging to \$INSTDIR\install.log will happen. \$INSTDIR must have a value before you call this function or it will not work. Note that the NSIS_CONFIG_LOG build setting must be set (scons NSIS_CONFIG_LOG=yes) when building (it is not set by default) to support this. See Building NSIS for more information about recompiling NSIS.

4.9.12.2 LogText

text

If installer logging is enabled, inserts text "text" into the log file.

```
IfFileExists $WINDIR\notepad.exe 0 +2
LogText "$$WINDIR\notepad.exe exists"
```

4.9.13 Section Management

4.9.13.1 SectionSetFlags

```
section_index section_flags
```

Sets the section's flags. The flag is a 32-bit integer. The first bit (lowest) represents whether the section is currently selected, the second bit

represents whether the section is a section group (don't modify this unless you really know what you are doing), the third bit represents whether the section is a section group end (again, don't modify), the fourth bit represents whether the section is shown in bold or not, the fifth bit represents whether the section is read-only, the sixth bit represents whether the section group is to be automatically expanded, the seventh bit is set for section groups which are partially selected, the eighth bit is internally used for partially selected section group toggling and the ninth bit is used for reflecting section name changes. The error flag will be set if an out of range section is specified.

Each flag has a name, prefixed with `SF_`:

```
!define SF_SELECTED 1
!define SF_SECGRP 2
!define SF_SECGRPEND 4
!define SF_BOLD 8
!define SF_RO 16
!define SF_EXPAND 32
!define SF_PSELECTED 64
```

For an example of usage please see the one-section.nsi example.

For more useful macros and definitions, see Include\Sections.nsh.

```
Section test test_section_id
SectionEnd

Function .onInit
    # set section 'test' as selected and read-only
    IntOp $0 ${SF_SELECTED} | ${SF_RO}
    SectionSetFlags ${test_section_id} $0

FunctionEnd
```

4.9.13.2 SectionGetFlags

```
section_index user_var(output)
```

Retrieves the section's flags. See SectionSetFlags for a description of the flags. The error flag will be set if an out of range section is specified.

```
Section test test_section_id
SectionEnd

Function .onSelChange
   # keep section 'test' selected
   SectionGetFlags ${test_section_id} $0
   IntOp $0 $0 | ${SF_SELECTED}
   SectionSetFlags ${test_section_id} $0
FunctionEnd
```

4.9.13.3 SectionSetText

```
section_index section_text
```

Sets the description for the section section_index. If the text is set to "" then the section will be hidden. The error flag will be set if an out of range section is specified.

```
Section "" test_section_id
SectionEnd

Function .onInit
  # change section's name to $WINDIR
  SectionSetText ${test_section_id} $WINDIR
FunctionEnd
```

4.9.13.4 SectionGetText

```
section_index user_var(output)
```

Stores the text description of the section section_index into the output. If the section is hidden, stores an empty string. The error flag will be set if an out of range section is specified.

```
Section test test_section_id
SectionEnd

Function .onInit
    # append $WINDIR to section's name
    SectionGetText ${test_section_id} $0
    StrCpy $0 "$0 - $WINDIR"
    SectionSetText ${test_section_id} $0
FunctionEnd
```

4.9.13.5 SectionSetInstTypes

```
section_index inst_types
```

Sets the install types the section specified by section_index defaults to the enabled state in. Note that the section index starts with zero. Every bit of inst_types is a flag that tells if the section is in that install type or not. For example, if you have 3 install types and you want the first section to be included in install types 1 and 3, then the command should look like this:

```
SectionSetInstTypes 0 5
```

because the binary value for 5 is "...00101". The error flag will be set if the section index specified is out of range.

```
Section test test_section_id
SectionEnd

Function .onInit
    # associate section 'test' with installation types 3
    SectionSetInstTypes ${test_section_id} 12
FunctionEnd
```

4.9.13.6 SectionGetInstTypes

```
section index user var(output)
```

Retrieves the install types flags array of a section. See above explanation about SectionSetInstTypes for a description of how to deal with the output. The error flag will be set if the section index is out of range.

```
Section test test_section_id
SectionEnd

Function .onInit
    # associate section 'test' with installation types 5
    SectionGetInstTypes ${test_section_id} $0
    IntOp $0 $0 | 16
    SectionSetInstTypes ${test_section_id} $0
FunctionEnd
```

4.9.13.7 SectionSetSize

```
section_index new_size
```

Sets the size of the section specified by section_index. Note that the index starts with zero. The Value for Size must be entered in KiloByte and supports only whole numbers.

```
Section test test_section_id
SectionEnd

Function .onInit
    # set required size of section 'test' to 100 bytes
    SectionSetSize ${test_section_id} 100

FunctionEnd
```

4.9.13.8 SectionGetSize

```
section index user var
```

Gets the size of the section specified by section_index and stores the value in the given user variable. Note that the index starts with zero. The error flag will be set if the section index is out of range.

```
Section test test_section_id
SectionEnd

Function .onInit
    # increase required size of section 'test' by 100 Ki
    SectionGetSize ${test_section_id} $0
    IntOp $0 $0 + 100
    SectionSetSize ${test_section_id} $0
FunctionEnd
```

4.9.13.9 SetCurInstType

```
inst_type_idx
```

Sets the current InstType. inst_type_idx should be between 0 and 31. The error flag is **not** set if an out of range InstType was used.

4.9.13.10 GetCurInstType

```
user_var
```

Get the current InstType and stores it in user_var. If the first install type is selected, 0 will be put in user_var. If the second install type is selected, 1 will be put in user_var, and so on. The value of \${NSIS_MAX_INST_TYPES} (32 by default) means that the user selected a custom set of sections (Simply selecting "Custom" in the drop-down menu is not enough to trigger this, the value is calculated by the sections actually selected).

4.9.13.11 InstTypeSetText

```
inst type idx text
```

Sets the text of the specified InstType. If the text is empty then the InstType is removed. By using a previously unused inst_type_idx number you can create new InstTypes. To add/remove Sections to this new InstType see SectionSetInstTypes. Unlike SectionIn the index is zero based, which means the first install type's index is 0.

```
InstType a
InstType b

Function .onInit
    # set first installation type's name to $WINDIR
    InstTypeSetText 0 $WINDIR
    # set second installation type's name to $TEMP
    InstTypeSetText 1 $TEMP
FunctionEnd
```

4.9.13.12 InstTypeGetText

```
inst_type_idx user_var
```

Gets the text of the specified InstType.

```
InstType a
InstType b

Function .onInit
   InstTypeGetText 0 $0
   DetailPrint $0 # prints 'a'
   InstTypeGetText 1 $0
   DetailPrint $0 # prints 'b'
FunctionEnd
```

4.9.14 User Interface Instructions

4.9.14.1 BringToFront

Makes the installer window visible and brings it to the top of the window list. If an application was executed that shows itself in front of the installer, BringToFront would bring the installer back in focus.

Recent Windows versions restrict the setting of foreground windows. If the user is working with another application during installation, the user may be notified using a different method.

4.9.14.2 CreateFont

```
user_var(handle output) face_name [height] [weight] [/
```

Creates a font and puts its handle into user_var. For more information about the different parameters have a look at MSDN's page about the Win32 API function CreateFont().

You can get the current font used by NSIS using the ^Font and ^FontSize LangStrings.

```
!include WinMessages.nsh
GetDlgItem $0 $HWNDPARENT 1
CreateFont $1 "Times New Roman" "7" "700" /UNDERLINE
SendMessage $0 ${WM_SETFONT} $1 1
```

4.9.14.3 DetailPrint

```
user_message
```

Adds the string "user_message" to the details view of the installer.

DetailPrint "this message will be shown in the install

4.9.14.4 EnableWindow

```
hwnd state (1|0)
```

Enables or disables mouse and keyboard input to the specified window or control. Possible states are 0 (disabled) or 1 (enabled).

```
GetDlgItem $0 $HWNDPARENT 1
EnableWindow $0 0
Sleep 1000
EnableWindow $0 1
```

4.9.14.5 FindWindow

```
user_var(hwnd output) windowclass [windowtitle] [windo
```

Searches for a window. Behaves like Win32's FindWindowEx(). Searches by windowclass (and/or windowtitle if specified). If windowparent or childafter are specified, the search will be restricted as such. If windowclass or windowtitle is specified as "", they will not be used for the search. If the window is not found the user variable is set to 0. To accomplish old-style FindWindow behavior, use FindWindow with SendMessage.

```
FindWindow $1 "#32770" "" $HWNDPARENT # Finds the inne FindWindow $2 "EDIT" "" $1 # Finds the first edit cont
```

4.9.14.6 GetDlgltem

```
user_var(output) dialog item_id
```

Retrieves the handle of a control identified by item_id in the specified dialog box dialog. If you want to get the handle of a control in the inner dialog, first use FindWindow to get the handle of the inner dialog.

GetDlgItem \$0 \$HWNDPARENT 1 # next/install button

4.9.14.7 HideWindow

Hides the installer window.

4.9.14.8 IsWindow

```
HWND jump_if_window [jump_if_not_window]
```

If HWND is a window, Gotos jump_if_window, otherwise, Gotos jump_if_not_window (if specified).

```
GetDlgItem $0 $HWNDPARENT 1
IsWindow $0 0 +3
   MessageBox MB_OK "found a window"
   Goto +2
   MessageBox MB_OK "no window"
```

4.9.14.9 LockWindow

on|off

LockWindow on prevents the main window from redrawing itself upon changes. When LockWindow off is used, all controls that weren't redrawn since LockWindow on will be redrawn. This makes the pages flickering look nicer because now it flickers a group of controls at the same time, instead of one control at a time. The individual control flickering is more noticeable on old computers.

4.9.14.10 SendMessage

```
HWND msg wparam lparam [user_var(return value)] [/TIME
```

Sends a message to HWND. If a user variable \$x is specified as the last parameter (or one before the last if you use /TIMEOUT), the return value from SendMessage will be stored in it. Note that when specifying 'msg' you must just use the integer value of the message. Include WinMessages.nsh to have all Windows messages defined in your script.

If you wish to send strings use "STR:a string" as wParam or IParam where needed. Use /TIMEOUT=time_in_ms to specify the duration, in milliseconds, of the time-out period.

```
!include WinMessages.nsh
FindWindow $0 "Winamp v1.x"
SendMessage $0 ${WM_CLOSE} 0 0

GetDlgItem $1 $HWNDPARENT 2
SendMessage $1 ${WM_SETTEXT} 0 "STR:Goodbye"
```

4.9.14.11 SetAutoClose

```
true|false
```

Overrides the default auto window-closing flag (specified for the installer using AutoCloseWindow, and false for the uninstaller). Specify 'true' to have the install window immediately disappear after the install has completed, or 'false' to make it require a manual close.

4.9.14.12 SetBrandingImage

```
[/IMGID=item id in dialog] [/RESIZETOFIT] path to bitm
```

Sets the current bitmap file displayed as the branding image. If no IMGID is specified, the first image control found will be used, or the image control created by AddBrandingImage. Note that this bitmap must be present on the user's machine. Use File first to put it there. If /RESIZETOFIT is specified the image will be automatically resized (very poorly) to the image control size. If you used AddBrandingImage you can get this size by compiling your script and watching for AddBrandingImage output, it will tell you the size. SetBrandingImage will not work when called from .onInit!

4.9.14.13 SetDetailsView

show|hide

Shows or hides the details, depending on which parameter you pass. Overrides the default details view, which is set via ShowInstDetails.

4.9.14.14 SetDetailsPrint

none|listonly|textonly|both|lastused

Sets mode at which commands print their status. None has commands be quiet, listonly has status text only added to the listbox, textonly has status text only printed to the status bar, and both enables both (the default). For extracting many small files, textonly is recommended (especially on win9x with smooth scrolling enabled).

SetDetailsPrint none File "secret file.dat" SetDetailsPrint both

4.9.14.15 SetCtlColors

```
hwnd [/BRANDING] [text color] [transparent|bg color]
```

Sets the text and background color of a static control, edit control, button or a dialog. *text_color* and *bg_color* don't accept variables. Use GetDlgItem to get the handle (HWND) of the control. To make the control transparent specify "transparent" as the background color value. You can also specify /BRANDING with or without text color and background color to make the control completely gray (or any other color you choose). This is used by the branding text control in the MUI.

```
FindWindow $0 "#32770" "" $HWNDPARENT GetDlgItem $0 $0 1006 SetCtlColors $0 0xFF0000 0x00FF00
```

Warning: Setting the background color of check boxes to "transparent"

may not function properly when using xpstyle on. The background may be completely black, instead of transparent, when using certain Windows themes.

4.9.14.16 SetSilent

```
silent | normal
```

Sets the installer to silent mode or normal mode. See SilentInstall for more information about silent installations. Can only be used in .onlnit.

4.9.14.17 **ShowWindow**

```
hwnd show state
```

Sets the visibility of a window. Possible show_states are the same as the Windows ShowWindow function. SW_* constants are defined in Include\WinMessages.nsh.

```
!include WinMessages.nsh
GetDlgItem $0 $HWNDPARENT 1
ShowWindow $0 ${SW_HIDE}
Sleep 1000
ShowWindow $0 ${SW_SHOW}
```

4.9.15 Multiple Languages Instructions

4.9.15.1 LoadLanguageFile

```
language_file.nlf
```

Loads a language file for the construction of a language table. All of the language files that ship with NSIS are in Contrib\Language Files

After you have inserted the language file \${LANG langfile} will be defined

as the language id (for example, \${LANG_ENGLISH} will be defined as 1033). Use it with LangString, LicenseLangString, LangDLL and VIAddVersionKey.

4.9.15.2 LangString

name language id 0 string

Defines a multilingual string. This means its value may be different (or not, it's up to you) for every language. It allows you to easily make your installer multilingual without the need to add massive switches to the script.

Each language string has a name that identifies it and a value for each language used by the installer. They can be used in any runtime string in the script. To use a language string all you need to add to the string is \$(LangString_name_here) where you want the LangString to be inserted.

Notes:

- Unlike defines that use curly braces {}, language strings use parenthesis ().
- If you change the language in the .onInit function, note that language strings in .onInit will still use the detected language based on the user's default Windows language because the language is initialized after .onInit.
- Always set language strings for every language in your script.
- If you set the language ID to 0 the last used language by LangString or LoadLanguageFile will be used.

Example of usage:

```
LangString message ${LANG_ENGLISH} "English message"
LangString message ${LANG_FRENCH} "French message"
LangString message ${LANG_KOREAN} "Korean message"

MessageBox MB OK "A translated message: $(message)"
```

4.9.15.3 LicenseLangString

```
name language id|0 license path
```

Does the same as LangString only it loads the string from a text/RTF file and defines a special LangString that can only be used by LicenseData.

```
LicenseLangString license ${LANG_ENGLISH} license-engl LicenseLangString license ${LANG_FRENCH} license-frenc LicenseLangString license ${LANG_GERMAN} license-germa LicenseData $(license)
```

4.10 Multiple Languages

As of version 2 NSIS fully supports multiple languages. The interface of one installer can support multiple languages.

Use LoadLanguageFile for every language to load the default interface texts and language properties.

The default interface texts can easily be changed using instructions like ComponentText etc.

You can also use the contents of the standard language strings in your own strings (for example, \$(^Name) contains the installer's name set using the Name instruction). The names of all standard language strings are listed as comments just above the strings in the language files. The language files are located in Contrib\Language Files.

To create your own language strings, use LangString.

For an example of an installer with multiple languages, see languages.nsi.

4.10.1 Language Selection

When the installer starts up it goes through these steps to select the interface language:

- 1. Get user's default Windows UI language
- 2. Find a perfect match for the language
- 3. If there is no perfect match, find a primary language match
- 4. If there is no match, use the first language defined in the script (make sure your first language is a common one like English)
- 5. If the language variable \$LANGUAGE has changed during .onInit, NSIS goes through steps 2 to 4 again.

4.10.2 LangDLL Plug-in

The LangDLL plug-in allows you to give the user an option to choose the language of the installer. Just push the language id (\${LANG_langfile}) and its name for every language in your installer, then the number of languages pushed, the caption, and the text that tells the user to select the language, call the plug-in function named LangDialog, pop the returned value into \$LANGUAGE and you're good to go. If the user clicks on the cancel button the return value will be "cancel".

For an example of usage see languages.nsi.

4.10.3 RTL Languages

RTL languages are languages that are written from right to left (e.g. Arabic and Hebrew). NSIS fully supports RTL languages. In the language file there is a place to specify if the language is RTL or not. To find out at runtime if the current language is RTL or not, check the value of the \$(^RTL) language string. It will be 1 if the language is RTL and 0 otherwise. This can be useful when using plug-ins that create dialogs, they usually have RTL settings too.

4.11 Plug-in DLLs

The abilities of the NSIS scripting language can be extended by utilising functionality provided in a DLL file. Probably the best known example of this is the InstallOptions.dll bundled with every NSIS release.

When the NSIS compiler starts it scans the plug-ins directory for DLLs and makes a list of the plug-ins found and their exported functions. During compilation, if a sequence such as fred::flintstone is encountered where the compiler expected to find a language keyword the compiler will look through this list. If a list entry specifies that fred.dll exports function flintstone NSIS will pack the fred.dll file into the created installer binary.

During execution of a plug-in command NSIS will unpack the necessary DLL to a temporary folder (\$PLUGINSDIR), push all of the arguments specified (right-to-left order), and then execute the DLL function.

4.11.1 Using Plug-in Commands

A plug-in call looks like this:

InstallOptions::dialog "ini file location.ini"

All parameters are pushed onto the stack (in this case, the plug-in function only needs one parameter). Some plug-in commands may not need any parameters on the stack, others might require more of them. To use a plug-in command you will need to read the documentation for the plug-in so that you know what parameters its functions require.

4.11.2 Calling plug-ins manually

If you want to call a plug-in that is stored on user's hard drive or somewhere else, use CallInstDLL. Almost all plug-ins provide installer functionality, so using plug-in commands is way easier. Using CallInstDLL can be useful when you have created plug-ins that are linked to a certain version of your application and are being copied to the installation folder.

4.12 Silent Installers/Uninstallers

Silent installers are installers which require no user intervention and have no user interface. The user doesn't see any dialog and isn't asked any questions. This is useful for network administrators who wish to install or uninstall something without user intervention so they can perform the operation quickly over any number of computers. It is also useful for other developers who wish to embed another installer in their own and collect all of the required information on their installer instead of showing two installers.

NSIS installers and uninstallers can be both silent and not silent. When an installer or an uninstaller is silent, not all callback functions are called. .onGUIInit, .onGUIEnd, their uninstaller equivalents and any callback related to a specific page or page type will not be called.

There are several methods to make an installer or an uninstaller silent:

- 1. SilentInstall and SilentUninstall
- 2. SetSilent
- 3. Passing /S on the command line (case sensitive)

To check if the installer/uninstaller is silent use IfSilent.

To make sure your installer will be silent when it needs to, you should check with IfSilent before each command that might require user intervention or create a window. The MessageBox command, which is the most common culprit in silent installers, has the /SD switch to set a default answer for silent installers. If you want your installer/uninstaller to be able to be completely silent you should use this switch. All internal NSIS message boxes have defaults for silent installers. The silent.nsi example demonstrates all aspects of this topic.

Since the directory page is not shown in silent installers the user has an option to specify the installation directory on the command line (this also works on non-silent installers/uninstallers). To do that, the user uses the /D switch as in the following example:

```
foo.exe /S /D=C:\Program Files\Foo
```

If your installer/uninstaller requires some more information that can not be gathered when silent, you can allow the user to specify that information on the command line and process it in .onInit. You can use GetOptions.

```
!include FileFunc.nsh
!insertmacro GetParameters
!insertmacro GetOptions

Function .onInit
   ${GetParameters} $R0
   ClearErrors
   ${GetOptions} $R0 /USERNAME= $0

FunctionEnd
```

The above example will copy the value the user passes on after /USERNAME= into \$0. This allows the user to specify the required information on the command line instead of using the interactive user interface. The user can use:

```
foo.exe /S /USERNAME=Bar /D=C:\Program Files\Foo
```

or:

or:

foo.exe /S /USERNAME=string with spaces /D=C:\Program

foo.exe /S /USERNAME="string with spaces" /D=C:\Progra

If your installer/uninstaller requires a lot of information and you want it to be able to be silent, you should allow the user to pass on a path to an answers file. This would be much more comfortable than writing all of the information on the command line.

5.1 Compiler Utility Commands

These commands are similar to the C preprocessor in terms of purpose and functionality. They allow file inclusion, conditional compilation, executable header packing and process execution during the build process. Note: None of these commands allow the use of variables.

Number literals support the 0b, 0o, 0n and 0x radix prefixes (base 2, 8, 10 and 16 respectively). Note: The deprecated plain 0 octal prefix is also supported in some places but its usage is discouraged.

5.1.1!include

[/NONFATAL] [/CHARSET=ACP|0EM|CP#|UTF8|UTF16LE|UTF16BE

This command will include 'file' as if it was part of the original script. Note that if a file is included in another directory, the current directory is still where the script was compiled from (not where the included file resides). If the compiler can't find the file it will look for it in every include directory. See !addincludedir for more information. If the /nonfatal switch is used and no files are found, a warning will be issued instead of an error. /charset can be used to specify a codepage for plain text files without a BOM.

```
!include WinMessages.nsh
!include Library.nsh
!include /CHARSET=CP1252 C:\MyConfig.nsi
!include ..\MyConfig.nsh
!include /NONFATAL file that may exist or not.nsh
```

5.1.2 !addincludedir

directory

Adds another include directory to the include directories list. This list is searched when !include is used. This list's initial value is

\${NSISDIR}\Include.

```
!addincludedir ..\include
!include something.nsh
```

5.1.3 !addplugindir

```
[/x86-ansi | /x86-unicode] directory
```

Causes the NSIS compiler to scan the given directory for plug-in DLLs. If you don't specify the plug-in architecture it is assumed to match the current target architecture. If the architecture does not match the installer will probably crash!

```
!addplugindir ..\myplugin
MyPlugin::SomeFunction
```

5.1.4 !appendfile

```
[/CHARSET=ACP|0EM|CP#|UTF8[SIG]|UTF16<LE|BE>[BOM]] [/R
```

Appends *text* to *file*. The text is written as ANSI (ACP) unless the file already has a BOM. Using /CHARSET will force a specific character encoding. \$\n will be translated to \$\r\$\n on Windows unless you specify /RawNL.

```
!tempfile FILE
!appendfile "${FILE}" "XPStyle on$\n"
!appendfile "${FILE}" "Name 'test'$\n"
!include "${FILE}"
!delfile "${FILE}"
!undef FILE
```

5.1.5 !cd

```
new_path
```

This command will change the compiler to the new directory, new_path.

new_path can be relative or absolute.

```
!cd ..\more-scripts\new
```

5.1.6 !delfile

```
[/nonfatal] file
```

This command deletes a file.

```
!tempfile FILE
!delfile "${FILE}"
!undef FILE
```

5.1.7 !echo

message

This command will echo a message to the user compiling the script.

```
!echo "hello world"
```

5.1.8 !error

[message]

This command will issue an error to the script compiler and will stop execution of the script. You can also add a message to this error.

```
!ifdef VERSION & NOVERSION
  !error "both VERSION and NOVERSION are defined"
!endif
```

5.1.9 !execute

```
command [compare comparevalue | symbol]
```

This command will execute 'command' using a call to CreateProcess().

Unlike !system, it does not use the command line processor, so input/output redirection and commands like 'cd', 'dir' and 'type' can not be used. Currently, the only known advantage of !execute over !system is that it does not give trouble when the current working directory is specified using UNC.

On POSIX platforms, !execute will use system() just like !system.

!execute '"\$%WINDIR%\notepad.exe" /P "\${NSISDIR}\COPYI

5.1.10 !makensis

```
parameters [compare comparevalue | symbol]
```

This command will !execute a new instance of MakeNSIS with the parameters you specify.

```
!makensis '-DGENERATEUNINST "${__FILE__}"' = 0
!system '"signtool" sign ...' = 0
```

5.1.11 !packhdr

```
tempfile command
```

This option makes the compiler use an external EXE packer (such as Petite or UPX) to compress the executable header. Specify a temporary file name (such as "temp.dat") and a command line (such as "C:\program files\upx\upx -9 temp.dat") to compress the header.

!packhdr "\$%TEMP%\exehead.tmp" '"C:\Program Files\UPX\

5.1.12 !finalize

```
command [compare comparevalue]
```

This option will execute 'command' using a call to system() after the output EXE has been generated. You can typically use it to sign (Authenticode) your installer. If 'command' contains a '%1' it will be

replaced by the executables filename.

```
!finalize 'sign.bat "%1" "Product Installer" http://ex
```

5.1.13 !system

```
command [compare comparevalue | symbol]
```

This command will execute 'command' using a call to system(). You can store the return value in a define ('symbol') or halt execution if the return value compared (using 'compare') to 'comparevalue' is false. 'compare' can be '<' or '>' or '<>' or '='.

```
!system '"%WINDIR%\notepad.exe" "${NSISDIR}\COPYING"'
!system 'echo !define something > newinclude.nsh'
!include newinclude.nsh
!ifdef something
   !echo "something is defined"
!endif
```

5.1.14 !tempfile

```
symbol
```

This command creates a temporary file. It puts its path into a define, named *symbol*.

```
!tempfile PACKHDRTEMP
!packhdr "${PACKHDRTEMP}" '"C:\Program Files\UPX\upx.e
!tempfile FILE
!define /date DATE "%H:%M:%S %d %b, %Y"
!system 'echo built on ${DATE} > "${FILE}"'
!undef DATE
File /oname=build.txt "${FILE}"
!delfile "${FILE}"
!undef FILE
```

5.1.15 !getdllversion

localfilename define_basename

This is similar to GetDLLVersionLocal, only it stores the version number in defines and can therefore be used anywhere, not just inside functions and sections.

```
!getdllversion "$%WINDIR%\Explorer.exe" expv_
!echo "Explorer.exe version is ${expv_1}.${expv_2}.${e
```

5.1.16 !warning

```
[message]
```

This command will issue a warning to the script compiler. You can also add a message to this warning.

```
!ifdef USE_DANGEROUS_STUFF
  !warning "using dangerous stuff"
!endif
```

5.1.17 !verbose

```
level | push | pop
```

This command will set the level of verbosity. 4=all, 3=no script, 2=no info, 1=no warnings, 0=none.

Passing push will cause !verbose to push the current verbosity level on a special stack. Passing pop will cause !verbose to pop the current verbosity level from the same stack and use it.

```
!verbose push
!verbose 1
!include WinMessages.nsh
!verbose pop
```

5.2 Predefines

You can use these standard predefines to automatically add the build time to the title of development versions, add the date to the version number, etc.

5.2.1 \${__COUNTER__}}

Expands to a number (Starting at 0 and incrementing by 1 every time it is used)

5.2.2 \${__FILE__}

Current script name.

5.2.3 \${___FILEDIR___}

Current script directory.

Current line number.

Date when the script started compiling according to the current locale.

Time when the script started compiling according to the current locale.

5.2.7 \${__TIMESTAMP__}}

Date & time of the last modification to the script file according to the current locale.

5.2.8 \${NSIS_VERSION}

NSIS version used to build the script.

5.2.9 \${NSIS_PACKEDVERSION}

NSIS version as a 32-bit number.

```
!if 0x3014000 >= "${NSIS_PACKEDVERSION}"
  !error "NSIS 3.15 or higher is required to build thi
!endif
```

5.2.10 \${NSIS_CHAR_SIZE}

The size of a character code unit (in bytes). 1 in ANSI installers and 2 in Unicode installers.

A grapheme cluster consists of a base character plus optional combining characters and diacritics and is defined as one or more code points. One or more code units is required to encode a single code point.

5.2.11 \${NSIS_PTR_SIZE}

The size of a pointer (in bytes) in the generated installer.

5.2.12 \${U+1}...\${U+10FFFF}

A Unicode (UCS-4) character.

DetailPrint "\${U+2115}SIS" # DOUBLE-STRUCK CAPITAL N +

5.2.13 Scope Predefines

Standard predefines that contain information about the current code scope.

5.2.13.1 \${__GLOBAL__}

Defined in the global scope.

```
Section test
  !ifdef ${__GLOBAL__}
    !error "this shouldn't be here!"
  !endif
SectionEnd

PageEx instfiles
  !ifdef ${__GLOBAL__}
    !error "this shouldn't be here!"
  !endif
PageExEnd
```

5.2.13.2 \${__SECTION__}}

Defined as the section name, without any prefixes, in section scope.

```
!ifdef SECTION
  !error "this shouldn't be here!"
!endif
Section test
  !ifndef SECTION
    !error "missing predefine!"
  !endif
  !if ${ SECTION } != test
    !error "wrong predefine value!"
  !endif
SectionEnd
Section !test
  !if ${ SECTION } != test
    !error "wrong predefine value!"
  !endif
SectionEnd
```

```
Section un.test
  !if ${__SECTION__}} != test
    !error "wrong predefine value!"
  !endif
SectionEnd
```

5.2.13.3 \${__FUNCTION__}

Defined as the function name, without any prefixes, in function scope.

```
!ifdef FUNCTION
  !error "this shouldn't be here!"
!endif
Function test
  !ifndef FUNCTION
    !error "missing predefine!"
  !endif
  !if ${ FUNCTION } != test
    !error "wrong predefine value!"
  !endif
FunctionEnd
Function un.test
  !if ${ FUNCTION } != test
    !error "wrong predefine value!"
  !endif
FunctionEnd
```

5.2.13.4 \${__PAGEEX__}

Defined as the page type in PageEx scope.

```
!ifdef __PAGEEX__
```

```
!error "this shouldn't be here!"
!endif

PageEx instfiles
  !ifndef __PAGEEX__
    !error "missing predefine!"
  !endif

!if ${__PAGEEX__} != instfiles
    !error "wrong page type"
  !endif
PageExEnd
```

5.2.13.5 \${__UNINSTALL__}}

Defined in section, function or PageEx scopes of the uninstaller.

```
!ifdef __UNINSTALL__
  !error "this shouldn't be here!"
!endif

Function test
  !ifdef __UNINSTALL__
    !error "this shouldn't be here!"
  !endif
FunctionEnd

Function un.test
  !ifndef __UNINSTALL__
    !error "missing predefine!"
  !endif
FunctionEnd
```

5.2.13.6 \${__MACRO__}}

Defined as the name of the current macro.

5.3 Read environment variables

5.3.1 \$%envVarName%

\$%envVarName% will be replaced at compile time by the environment variable envVarName.

5.4 Conditional Compilation

The compiler maintains a list of defined symbols, which can be defined using !define or the /D command line switch. These defined symbols can be used for conditional compilation (using !ifdef) or for symbol replacement (a simple form of macros). To replace a symbol with its value, use \${SYMBOL} (if SYMBOL is not defined, no translation will occur). The translation is first-come-first-served, meaning if you do:

```
!define symbol one ${symbol two}
```

If symbol_two is defined when that line occurs, it will be replaced. Otherwise, any replacing will occur when \${symbol_one} is referenced.

Define/conditional compilation related commands:

5.4.1 !define

```
[/ifndef | /redef] ([/date|/utcdate] gflag [value]) |
```

This command will add *gflag* to the global define list. This will have a similar effect as using the /D switch on the command line (the define only becomes effective after the !define command).

If /date or /utcdate are used, value will be passed to strftime() and the result will be used as the value of gflag. strftime converts special symbols into certain parts of the current time or date. For example, %H will be converted into the current hour in 24-hour format. For a complete list of available symbols, search for strftime on MSDN. On POSIX, you can get the list by using man strftime.

If /math is used, the result of 'val1 OP val2', where OP may be $+,-,*,\&,|,^,/,<<,>>>$ or %, will be used as the value of gflag. Note that val1 AND val2 MUST be integer values!

If */file* is used, the entire text file specified (including whitespace and newlines) will be read and stuffed into *gflag*.

```
!define USE_SOMETHING
!define VERSION 1.2
!define /date NOW "%H:%M:%S %d %b, %Y"
!define /math RESULT 3 + 10
!define /math REST 15 % ${RESULT}
!define /file BunchaStuff somesourcefile.cpp
!define /redef USE_SOMETHING ${RESULT} ;redefine USE_S
```

5.4.2 !undef

```
gflag
```

Removes an item from the global define list. Note that \${SYMBOL} where SYMBOL is undefined will be translated to "\${SYMBOL}".

```
!define SOMETHING
!undef SOMETHING
```

5.4.3 !ifdef

```
gflag [bcheck gflag [...]]
```

This command, when paired with an !endif command, will tell the compiler whether or not to compile the lines in between the two lines. If gflag is globally defined (using !define or the /D switch), then the contained lines will be compiled. Otherwise, they will be skipped. 'bcheck' can be specified as & (boolean and) or | (boolean or) along with more gflags -- precedence is simple, left to right.

```
!define SOMETHING
!ifdef SOMETHING
  !echo "SOMETHING is defined"
!endif
!undef SOMETHING
!ifdef SOMETHING
  !echo "SOMETHING
    !echo "SOMETHING is defined" # will never be printed
!endif
```

5.4.4 !ifndef

```
gflag [bcheck gflag [...]]]
```

The opposite of !ifdef. The lines will be compiled when the gflag has not been defined.

5.4.5 !if

```
[!] value [op value2]
[!] /FileExists "c:\path\file.exe"
```

This command, when paired with an !endif command, will tell the compiler whether or not to compile the lines in between the two lines. If value is non-zero, or the comparison of value and value2 depending on the operator results in true, the contained lines will be compiled. Otherwise, they will be skipped. op can be either == or != (case-insensitive string comparison), S== or S!= (case-sensitive string comparison), =, <>, <=, <, > or >= (int/hex/float comparison), & (bitwise AND comparison), && or || (boolean comparison). If [!] is set, the result will be flipped from true to false and vice versa.

```
!if 1 < 0x2
  !echo "1 is smaller than 2!!"
!else if ! 3.1 > 1.99
  !error "this line should never appear"
!else
  !error "neither should this"
!endif
!if /FileExists ".\cert.pfx"
```

5.4.6 !ifmacrodef

!endif

!finalize '".\sign.bat" "%1"'

```
gflag [bcheck gflag [...]]]
```

This command, when paired with an !endif command, will tell the compiler whether or not to compile the lines in between the two lines. If the macro gflag exists, then the contained lines will be compiled. Otherwise, they will be skipped. 'bcheck' can be specified as & (boolean and) or | (boolean or) along with more gflags -- precedence is simple, left to right.

```
!macro SomeMacro
!macroend
!ifmacrodef SomeMacro
    !echo "SomeMacro is defined"
!endif
```

5.4.7 !ifmacrondef

```
gflag [bcheck gflag [...]]]
```

The opposite of !ifmacrodef. The lines will be compiled when the macro gflag does not exist.

5.4.8 !else

```
[if|ifdef|ifndef|ifmacrodef|ifmacrondef [...]]
```

This command allows to easily insert different code when different defines or macros are set. You can create blocks like !ifdef/!else/!endif, !ifdef/!else ifdef/!else/!endif etc.

```
!ifdef VERSION
OutFile installer-${VERSION}.exe
!else
OutFile installer.exe
!endif
```

5.4.9 !endif

This command closes a block started with !if, !ifdef, !ifndef, !ifmacrodef or

!ifmacrondef.

5.4.10 !insertmacro

```
macro_name [parameter] [...]
```

Inserts the contents of a macro that was created with !macro. If the macro was created with parameters, then you must pass as many parameters to the macro as it requires.

```
!macro Print text
  DetailPrint "${text}"
!macroend
!insertmacro Print "some text"
!insertmacro Print "some more text"
```

5.4.11 !macro

```
macro_name [parameter][...]
```

Creates a macro named 'macro_name'. All lines between the !macro and the !macroend will be saved. To insert the macro later on, use !insertmacro. !macro definitions can have one or more parameters defined. The parameters may be accessed the same way a !define would (e.g. \${PARMNAME}) from inside the macro.

```
!macro SomeMacro parm1 parm2 parm3
  DetailPrint "${parm1}"
  MessageBox MB_OK "${parm2}"
  File "${parm3}"
!macroend
```

5.4.12 !macroend

Ends a macro that was started with !macro.

5.4.13 !macroundef

macro name

Deletes a macro.

5.4.14 !searchparse

```
[/ignorecase] [/noerrors] [/file] source_string_or_fil
```

Parses *source_string_or_file* (which is treated as a string, or as a filename if */file* is set), looking for *substring_start*. If *substring_start* is found, then *OUTPUTSYMBOL1* is defined to the rest of the string (minus any other *substring* that may be found). Any number of *OUTPUTSYMBOLx* may be specified, and the final *substring* is optional.

If */noerrors* is specified, matching less than the full number of strings is allowed (all *OUTPUTSYMBOLx* after the not-found substring will be ignored).

If /file is specified, the file is treated as a series of lines. The file is searched until all substrings are matched. If /noerrors is specified and not all strings are matched, the first line with the most symbols matched is used.

search filename.cpp for a line '#define APP_VERSION
!searchparse /file filename.cpp `#define APP_VERSION "

5.4.15 !searchreplace

```
[/ignorecase] symbol_out source_string searchfor repla
```

Searches *source_string*, looking for *searchfor* and replacing all instances of it with *replacewith*. Unlike !define, !searchreplace allows you to redefine *symbol_out* without warning or error.

```
# defines ${blah} to "i like ponies"
!searchreplace blah "i love ponies" "love" "like"
```

Appendix A: Modern User Interface

NSIS 2 makes it is possible to create installers with a custom user interface. The Modern UI is a interface with a style like the wizards of recent Windows versions. This new interface also features new pages (Welcome, Finish, Start Menu) and a description area on the components page. The interface and the graphics can be customized using the provided settings. Using the Modern UI macros and language files, writing scripts with a modern interface is easy.

For more information and documentation see the Modern UI 2 Readme.

NSIS 2.34 brought with it a new version of Modern UI - version 2. It is faster and more extendible. It allows plug-ins to add new types of pages and even change existing pages using a simple NSH file. It also uses nsDialogs which faster than its elder sibling - InstallOptions.

For more information and documentation of the old version see the Modern UI Readme.

Appendix B: DLL/TLB Library Setup

- Introduction
- Library Installation
 - Introduction
 - Parameters
 - Options
 - Notes
 - Example
- Library Uninstallation
 - Introduction
 - Parameters
 - Options
 - Example
- Visual Basic 6 Run-Time Files

B.1 Introduction

The Library header file can be used to setup dynamic link libraries (DLL) and type libraries (TLB). If necessary, the following actions will be performed:

- File copying
- File copying on reboot
- Version checks
- Registration and unregistration
- Registration and unregistration on reboot
- Shared DLL counting
- Windows File Protection checks

The macros are stored in the header file Library.nsh, which should be included in scripts using this system:

!include Library.nsh

Note that the library macros are limited on non-Windows platforms. DLL version information is required when compiling on non-Windows platforms.

B.2 Library Installation

B.2.1 Introduction

The InstallLib macro allows you to install a library. It sets the error flag if something went wrong during library setup.

To ask the user for a reboot, if required, use the Modern UI with a Finish page or use IfRebootFlag and make your own page or message box.

B.2.2 Parameters

libtype shared install localfile destfile tempbasedir

libtype

The type of the library

DLL - Dynamic link library (DLL)

REGDLL - DLL that has to be registered

REGEXE - EXE COM server that has to be registered using /regserver

TLB - Type library or DLL that contains a type library

REGDLLTLB - DLL that has to be registered and contains a type library

shared

Specify whether the library is shared with other applications

NOTSHARED - The library is not shared

\$VARNAME - Variable that is empty when the application is installed for the first time, which is when the shared library count will be increased.

install

Specify the installation method

REBOOT PROTECTED

- Upgrade the library on reboot when in use (required for system files).
- Upgrade the library if the file is not protected by Windows File Protection.

NOREBOOT PROTECTED

- Warns the user when the library is in use. The user will have to close applications using the library.
- Upgrade the library if the file is not protected by Windows File Protection.

REBOOT_NOTPROTECTED

- Upgrade the library on reboot when in use (required for system files).
- Upgrade the library without checking for Windows File Protection.

NOREBOOT_NOTPROTECTED

- Warns the user when the library is in use. The user will have to close applications using the library.
- Upgrade the library without checking for Windows File Protection.

localfile

Location of the library on the compiler system

destfile

Location to store the library on the user's system

tempbasedir

Directory on the user's system to store a temporary file when the system has to be rebooted.

For Windows 9x/ME support, this directory should be on the same volume as the destination file (destfile). The Windows temp directory could be located on any volume, so you cannot use this directory.

B.2.3 Options

Define any of the following before inserting a InstallLib macro to modify its behavior as specified.

B.2.3.1 LIBRARY_X64

- Installs a DLL built for Windows x64.
- Warning: This resets file system redirection.

B.2.3.2 LIBRARY_SHELL_EXTENSION

- Define this before inserting InstallLib macro to call SHChangeNotify with SHCNE ASSOCCHANGED after registration.
- Use this to refresh the shell when installing a shell extension or when changing file associations.

B.2.3.3 LIBRARY_COM

- Define this before inserting InstallLib macro to call CoFreeUnusedLibraries after registration.
- Use this for unloading all unnecessary libraries from memory when installing COM libraries.

B.2.3.4 LIBRARY_IGNORE_VERSION

- Define this before inserting InstallLib macro to ignore version information in the file and always install it, even if it already exists.
- Use this when an older or specific version is required.
- Not recommended for DLLs installed to \$SYSDIR.

B.2.4 Notes

If you need to support Windows 9x/ME, you can only use short

filenames (8.3).

• **Warning:** Always use redistributable files when deploying DLLs, never copy files from your system directory!

B.2.5 Example

B.2.5.1 Unshared DLL

!insertmacro InstallLib REGDLL NOTSHARED REBOOT NOTPR

B.2.5.2 Shared DLL

```
;Add code here that sets $ALREADY_INSTALLED to a non-
;already installed. For example:
IfFileExists "$INSTDIR\MyApp.exe" 0 new_installation
  StrCpy $ALREADY_INSTALLED 1
new_installation:
```

!insertmacro InstallLib REGDLL \$ALREADY_INSTALLED REB

B.3 Library Uninstallation

B.3.1 Introduction

The UnInstallLib macro allows you to uninstall a library. It sets the error flag if something went wrong during library removal.

B.3.2 Parameters

libtype shared uninstall file

libtype

The type of the library

DLL - Dynamic link library (DLL)
REGDLL - DLL that has to be unregistered
REGEXE - EXE COM server that has to be unregistered using
/unregserver
TLB - Type library or DLL that contains a type library
REGDLLTLB - DLL that has to be unregistered and contains a type library

shared

Specify whether the library is shared with other applications

NOTSHARED - The library is not shared SHARED - The library is shared and should be removed if the shared library count indicates that the file is not in use anymore..

uninstall

Specify the uninstallation method

NOREMOVE

 The library should not be removed. You should use this option for common or important system files such as the Visual Basic/C++/MFC runtimes.

REBOOT PROTECTED

- Remove the library on reboot when in use (required for system files).
- Remove the library if the file is not protected by Windows File Protection.

NOREBOOT_PROTECTED

- Warns the user when the library is in use. The user will have to close applications using the library.
- Remove the library if the file is not protected by Windows File Protection.

REBOOT_NOTPROTECTED

- Remove the library on reboot when in use (required for system files).
- Remove the library without checking for Windows File Protection.

NOREBOOT NOTPROTECTED

- Warns the user when the library is in use. The user will have to close applications using the library.
- Remove the library without checking for Windows File Protection.

file

Location of the library

B.3.3 Options

Define any of the following before inserting a UnInstallLib macro to modify its behavior as specified.

B.3.3.1 LIBRARY X64

- Uninstalls a DLL built for Windows x64.
- Warning: This resets SetRegView and file system redirection.

B.3.3.2 LIBRARY SHELL EXTENSION

 Define this before inserting UninstallLib macro to call SHChangeNotify with SHCNE_ASSOCCHANGED after unregistration. Use this to refresh the shell when uninstalling a shell extension or when changing file associations.

B.3.3.3 LIBRARY_COM

 Define this before inserting UninstallLib macro to call CoFreeUnusedLibraries after unregistration. Use this for unloading all unnecessary libraries from memory when uninstalling COM libraries.

B.3.4 Example

!insertmacro UnInstallLib REGDLL SHARED REBOOT_NOTPRO

B.4 Visual Basic 6 Run-Time Files

A new VB6RunTime.nsh header file is available for the setup of the VB6 run-time files. To obtain the latest run-time files, download vb6runtime.zip and extract this file.

```
!include VB6RunTime.nsh

Var AlreadyInstalled

Section "-Install VB6 run-time files"

;Add code here that sets $AlreadyInstalled to a non
IfFileExists "$INSTDIR\MyApp.exe" 0 new_installatio
    StrCpy $AlreadyInstalled 1
    new_installation:
    !insertmacro VB6RunTimeInstall C:\vb6runtimes $Alre

SectionEnd

Section "-un.Uninstall VB6 run-time files"
    !insertmacro VB6RunTimeUnInstall

SectionEnd
```

Remarks:

- You may have to install additional files for such Visual Basic application to work, such as OCX files for user interface controls.
- Installation of the run-time files requires Administrator or Power User privileges. Use the Multi-User header file to verify whether these privileges are available.
- Add a Modern UI finish page or another check (see IfRebootFlag) to allow the user to restart the computer when necessary.

Appendix C: Useful Scripts

- Get Internet Explorer version
- Is .NET Framework installed?
- Is Macromedia Flash Player installed?
- Connect to the Internet
- Get Installer Filename
- Prevent Multiple Instances
- More

C.1 Get Internet Explorer version

```
: GetIEVersion
; Based on Yazno's function, http://yazno.tripod.com/
; Returns 1-6 (IE Version) or '' (IE is not installed
; Usage:
  Call GetIEVersion
; Pop $R0 ; at this point $R0 is "5" or whatnot
Function GetIEVersion
Push $R0
 ClearErrors
 ReadRegStr $R0 HKLM "Software\Microsoft\Internet Ex
 IfErrors lbl 123 lbl 456
 lbl 456: : ie 4+
   Strcpy $R0 $R0 1
 Goto lbl done
 lbl 123: : older ie version
    ClearErrors
   ReadRegStr $R0 HKLM "Software\Microsoft\Internet
   IfErrors lbl_error
     StrCpy $R0 $R0 3
      StrCmp $R0 '100' lbl_ie1
     StrCmp $R0 '101' lbl ie2
      StrCmp $R0 '102' lbl ie2
      StrCpy $R0 '3'; default to ie3 if not 100, 101
      Goto lbl done
       lbl ie1:
          StrCpy $R0 '1'
       Goto lbl done
```

```
lbl_ie2:
        StrCpy $R0 '2'
        Goto lbl_done
        lbl_error:
            StrCpy $R0 ''
        lbl_done:
        Exch $R0
FunctionEnd
```

C.2 Is .NET Framework installed?

```
IsDotNETInstalled
; Based on GetDotNETVersion
   http://nsis.sourceforge.net/Get_.NET_Version
; Usage:
    Call IsDotNETInstalled
   Pop $0
    StrCmp $0 1 found_dotNETFramework no_dotNETFramew
Function IsDotNETInstalled
 Push $0
 Push $1
 StrCpy $0 1
 System::Call "mscoree::GetCORVersion(w, i ${NSIS_MA
 StrCmp $1 0 +2
    StrCpy $0 0
 Pop $1
 Exch $0
FunctionEnd
```

C.3 Is Macromedia Flash Player installed?

```
: IsFlashInstalled
; By Yazno, http://yazno.tripod.com/powerpimpit/
; Returns the result on top of the stack
; Usage:
    Call IsFlashInstalled
    Pop $R0; $R0 is "1" or "0" at this point
Function IsFlashInstalled
 Push $R0
ClearErrors
 ReadRegStr $R0 HKCR "CLSID\{D27CDB6E-AE6D-11cf-96B8-
 IfErrors lbl na
   StrCpy $R0 1
Goto lbl end
 lbl na:
   StrCpy $R0 0
lbl end:
Exch $R0
FunctionEnd
```

C.4 Connect to the Internet

```
; ConnectInternet (uses Dialer plug-in)
; Written by Joost Verburg
; This function attempts to make a connection to the
; connection available. If you are not sure that a sy
; has an active internet connection, call this functi
: files with NSISdl.
; The function requires Internet Explorer 3, but asks
; IE3 is not installed.
Function ConnectInternet
 Push $R0
    ClearErrors
   Dialer::AttemptConnect
    IfFrrors noie3
    Pop $R0
    StrCmp $R0 "online" connected
     MessageBox MB OK|MB ICONSTOP "Cannot connect to
     Quit ; This will quit the installer. You might w
    noie3:
    : IE3 not installed
   MessageBox MB OK | MB ICONINFORMATION "Please conne
    connected:
 Pop $R0
FunctionEnd
```

C.5 Get Installer Filename

```
System::Call 'kernel32::GetModuleFileName(p 0, t .R0,
;$R0 will contain the installer filename
```

C.6 Prevent Multiple Instances

Put the following code in your .onInit function:

```
System::Call 'kernel32::CreateMutex(p 0, i 0, t "myMu
Pop $R0

StrCmp $R0 0 +3
   MessageBox MB_OK|MB_ICONEXCLAMATION "The installer
Abort
```

'myMutex' must be replaced by a unique value!

C.7 More

You can find more useful scripts on the NSIS Wiki, the NSIS forum and the NSIS development page.

Appendix D: Useful Information

- Error Levels
- Add uninstall information to Add/Remove Programs
- Calling an external DLL using the System.dll plug-in
- Dump Content of Log Window to File
- How to Read REG_MULTI_SZ Values
- Predefined Macros for Unicode support

D.1 Error Levels

Like other applications, installers made by NSIS return error levels as a result of their execution. Checking the error level can be useful if you call an NSIS installer from another application or installer.

- 0 Normal execution (no error)
- 1 Installation aborted by user (cancel button)
- 2 Installation aborted by script

As of NSIS 2.01, you can set the error level to other values using SetErrorLevel.

Note that uninstallers copy themselves to the temporary directory and execute from there so the original uninstaller can be deleted. This means the error level the uninstaller sets is not available to the executing process, unless it simulates this copy process and executes the copied uninstaller. To simulate this process, use:

InitPluginsDir
CopyFiles \$INSTDIR\uninstaller.exe \$PLUGINSDIR
ExecWait '"\$PLUGINSDIR\uninstaller.exe" _?=\$INSTDIR' \$
DetailPrint "uninstaller set error level \$0"

If you don't do this, you'll only be able to know if the uninstaller failed copying itself to the temporary directory.

D.2 Add uninstall information to Add/Remove Programs

Create a key with your product name under HKLM\Software\Microsoft\Windows\CurrentVersion\Uninstall to add entries to the "Add/Remove Programs" section in the Control Panel. For Windows NT (NT4/2000/XP), it's also possible to create the key in the HKCU hive, so it will only appear for the current user. There are several values you can write to key to give information about your application and the uninstaller. Write a value using the WriteRegStr command (for strings) or WriteRegDWORD command (for DWORD values).

Example:

WriteRegStr HKLM "Software\Microsoft\Windows\CurrentVe
WriteRegStr HKLM "Software\Microsoft\Windows\CurrentVe

Required values

DisplayName (string) - Name of the application UninstallString (string) - Path and filename of the uninstaller. You should always quote the path.

Optional values

Some of the following values will not be used by older Windows versions.

InstallLocation (string) - Installation directory (\$INSTDIR)

DisplayIcon (string) - Path, filename and index of the icon that will be displayed next to your application name

Publisher (string) - (Company) name of the publisher

ModifyPath (string) - Path and filename of the application modify program InstallSource (string) - Location where the application was installed from

ProductID (string) - Product ID of the application

RegOwner (string) - Registered owner of the application RegCompany (string) - Registered company of the application

HelpLink (string) - Link to the support website HelpTelephone (string) - Telephone number for support

URLUpdateInfo (string) - Link to the website for application updates URLInfoAbout (string) - Link to the application home page

DisplayVersion (string) - Displayed version of the application VersionMajor (DWORD) - Major version number of the application VersionMinor (DWORD) - Minor version number of the application

NoModify (DWORD) - 1 if uninstaller has no option to modify the installed application

NoRepair (DWORD) - 1 if the uninstaller has no option to repair the installation

If both NoModify and NoRepair are set to 1, the button displays "Remove" instead of "Modify/Remove".

D.3 Calling an external DLL using the System.dll plug-in

Some installers need to call functions in third-party DLLs. A prime example of this is when installing a Palm(TM) conduit.

Some background about System.dll The System.dll plug-in enables calling of external DLLs by using its 'Call' function. There are a number of other functions provided by System.dll but they will not be covered here. For more details about the other functions, lock the doors, take the phone off the hook, screw your head on *real* tight and head on over to the System readme.

Data Types

System.dll recognises the following data types:

- v void (generally for return)
- p pointer (includes void*, HANDLE, HWND, UINT_PTR and so on)
- i int (a 32bit integer)
- I large integer (also known as int64)
- t text, string (LPTSTR, pointer to first character)
- k callback. See Callback section in system.html.
- * pointer specifier -> the proc needs the pointer to type, affects next char (parameter) [ex: '*i' pointer to int]

Mapping System.dll variables to NSIS script variables

There's not much point in being able to call an external function if you can't get any data back. System.dll maps function variables to NSIS script variables in the following way:

NSIS \$0..\$9 becomes System.dll r0..r9 NSIS \$R0..\$R9 becomes System.dll r10..r19

Each parameter is specified by type, input and output. To skip input or output use a dot. Examples:

String (pointer to a character array), input is 'happy calling':

```
t 'happy calling'
```

String (pointer to a character array), input is taken from \$5 and changes to the array made by the callee are saved into \$R8:

```
t r5R8
```

Pointer to an integer, value taken from \$1 and put into \$2:

```
*i r1r2
```

Pointer to a 64-bit integer, output pushed on stack, no input:

```
*l .s
```

Using System.dll::Call To call a function in a third party DLL, the Call function is used like this:

```
System::Call 'YourDllName::YourDllFunction(i, *i, t) i
```

The '(r0, .r1, r2) .r3' section at the end are the parameters that are passed between your DLL and your NSIS script. As can be seen in this parameters list, type and input/output can be separated. Each block of " (parms list) return value" overrides and/or adds to the last one. In this case, the first block specifies the types and the second specifies input and output.

Before starting to code the NSIS script

Before you start to code any NSIS code you need to know the full prototype of the function you are going to call. For the purposes of this example, we will use the 'CmGetHotSyncExecPath' function from the Palm 'CondMgr.dll'. This function is used to return the full path of 'HotSync.exe'.

Function Definition

int __stdcall CmGetHotSyncExecPath(TCHAR *pPath, int *piSize);

where

- pPath is a pointer to a character buffer. Upon return, this is the path
 & file name of the installed HotSync manager.
- piSize is a pointer to an integer that specifies the size (in TCHAR's), of the buffer referenced by the pPath parameter.

return values:

- 0: No error
- -1: A non-specific error occurred
- ERR_REGISTRY_ACCESS(-1006):Unable to access the Palm configuration entries
- ERR_BUFFER_TOO_SMALL(-1010): The buffer is too small to hold the requested information
- ERR_INVALID_POINTER(-1013):The specified pointer is not a valid pointer

Also, if the buffer is too small the value in *int is the size (in TCHARs) that the buffer should be.

This function definition maps to the following System.dll definition:

CmGetHotSyncExecPath(t, *i) i

i.e. It takes a text variable, a pointer to int, and returns an int value.

Using the external dll function

Now that we've sorted out what the function does and how it maps to the System.dll format we can use the function in a NSIS script.

First you have to change the output directory to that where the DLL you want to use is. It may also work if the DLL is in the system path but this hasn't been tested.

The following code fragment will install 'condmgr.dll' to a temporary directory, execute the CmGetHotSyncExecPath function and display returned data. Save this script

```
; **** snip ****
Function loadDll

SetOutPath $TEMP\eInspect ; create temp
File bin\CondMgr.dll ; copy dll the
StrCpy $1 ${NSIS_MAX_STRLEN} ; assign memor
System::Call 'CondMgr::CmGetHotSyncExecPath(t, *i) i
DetailPrint 'Path: "$0"'
DetailPrint "Path length: $1"
DetailPrint "Return value: $2"

FunctionEnd
; **** snip ****
```

and this function produces the following output in the 'details' page:

Output folder: c:\windows\TEMP\eInspect

Extract: CondMgr.dll

Path: "C:\Dave\palm\Hotsync.exe"

Path length: 24 Return value: 0

Written by djc

Acknowledgements & Thanks

Lots of thanks go to **kichik** and **Sunjammer** for spending a lot of time assisting in solving this problem. Also to **brainsucker** for creating the System.dll plug-in in the first place. Good Luck!

D.4 Dump Content of Log Window to File

This function will dump the log of the installer (installer details) to a file of your choice.

To use it, push a file name and call it. It will dump the log to the file specified. For example:

```
GetTempFileName $0
Push $0
Call DumpLog
```

Here is the function:

```
!define LVM GETITEMCOUNT 0x1004
!define LVM GETITEMTEXTA 0x102D
Function DumpLog # Written by KiCHiK
 Exch $5
 Push $0
 Push $1
 Push $2
 Push $3
 Push $4
 Push $6
  FindWindow $0 "#32770" "" $HWNDPARENT
 GetDlgItem $0 $0 1016
 StrCmp $0 0 error
  FileOpen $5 $5 "w"
 StrCmp $5 0 error
   SendMessage $0 ${LVM GETITEMCOUNT} 0 0 $6
   System::StrAlloc ${NSIS MAX STRLEN}
   Pop $3
   StrCpy $2 0
   System::Call "*(i, i, i, i, i, i, i, i, i) p \
```

```
(0, 0, 0, 0, 0, r3, ${NSIS MAX STRLEN}) .r1"
    loop: StrCmp $2 $6 done
      System::Call "User32::SendMessageA(p, i, p, p) i
        ($0, ${LVM GETITEMTEXTA}, $2, r1)"
      System::Call "*$3(&t${NSIS_MAX_STRLEN} .r4)"
      FileWrite $5 "$4$\r$\n"
      Int0p $2 $2 + 1
      Goto loop
    done:
      FileClose $5
      System::Free $1
      System::Free $3
      Goto exit
 error:
   MessageBox MB OK error
  exit:
    Pop $6
   Pop $4
   Pop $3
   Pop $2
   Pop $1
   Pop $0
    Exch $5
FunctionEnd
```

Here's the function to generate a UTF-16LE file if you're building a Unicode installer.

```
!define LVM_GETITEMCOUNT 0x1004
!define LVM_GETITEMTEXTW 0x1073

Function DumpLog # Written by KiCHiK, modified by Jim Exch $5
  Push $0
  Push $1
  Push $2
  Push $3
```

```
Push $4
Push $6
FindWindow $0 "#32770" "" $HWNDPARENT
GetDlgItem $0 $0 1016
StrCmp $0 0 error
FileOpen $5 $5 "w"
FileWriteWord $5 Oxfeff; Write the BOM
StrCmp $5 0 error
  SendMessage $0 ${LVM GETITEMCOUNT} 0 0 $6
 System::StrAlloc ${NSIS MAX STRLEN}
  Pop $3
 StrCpy $2 0
 System::Call "*(i, i, i, i, i, i, i, i, i) p \
    (0, 0, 0, 0, 0, r3, ${NSIS MAX STRLEN}) .r1"
 loop: StrCmp $2 $6 done
    System::Call "User32::SendMessageW(p, i, p, p) i
      ($0, ${LVM GETITEMTEXTW}, $2, r1)"
    System::Call "*$3(&t${NSIS MAX STRLEN} .r4)"
    FileWriteUTF16LE $5 "$4$\r$\n"
    Int0p $2 $2 + 1
    Goto loop
  done:
    FileClose $5
    System::Free $1
    System::Free $3
    Goto exit
error:
 MessageBox MB OK error
exit:
  Pop $6
  Pop $4
 Pop $3
 Pop $2
 Pop $1
 Pop $0
  Exch $5
```

FunctionEnd

D.5 How to Read REG_MULTI_SZ Values

KiCHiK wrote this script to help rpetges in this forum thread. It reads a registry value of the type REG_MULTI_SZ and prints it out. Don't forget to edit where it says "Edit this!" when you test this script. The values must point to a REG MULTI_SZ value or the example will spit out an error.

```
OutFile "REG MULTI SZ Reader.exe"
Name "REG MULTI SZ Reader"
ShowInstDetails show
!define HKEY CLASSES ROOT
                                   0x80000000
!define HKEY CURRENT USER
                                   0x80000001
!define HKEY LOCAL MACHINE
                                   0x80000002
!define HKEY USERS
                                   0x80000003
!define HKEY PERFORMANCE DATA
                                   0x80000004
!define HKEY PERFORMANCE TEXT
                                   0x80000050
!define HKEY PERFORMANCE NLSTEXT
                                  0x80000060
!define HKEY CURRENT CONFIG
                                   0x80000005
!define HKEY DYN DATA
                                   0x80000006
!define KEY QUERY VALUE
                                   0 \times 0001
!define KEY ENUMERATE SUB KEYS
                                   8000x0
!define REG NONE
                                   0
!define REG SZ
                                   1
!define REG EXPAND SZ
                                   2
                                   3
!define REG BINARY
                                   4
!define REG DWORD
!define REG DWORD LITTLE ENDIAN
                                   4
                                   5
!define REG DWORD BIG ENDIAN
!define REG LINK
                                   6
!define REG_MULTI_SZ
                                   7
!define RegOpenKeyEx
                          "Advapi32::RegOpenKeyExA(i, t
!define RegOueryValueEx
                          "Advapi32::RegOuervValueExA(i
!define RegCloseKey
                          "Advapi32::RegCloseKeyA(i) i"
```

```
###### Edit this!
!define ROOT KEY
                         ${HKEY CURRENT USER}
!define SUB KEY
                        "Software\Joe Software"
!define VALUE
                         "Strings"
###### Stop editing
Section "Read"
  StrCpy $0 ""
  StrCpy $1 ""
  StrCpy $2
  StrCpy $3 ""
  System::Call "${RegOpenKeyEx}(${ROOT KEY}, '${SUB KE
    0, ${KEY QUERY VALUE}|${KEY ENUMERATE SUB KEYS}, .
  StrCmp $3 0 goon
    MessageBox MB OK|MB ICONSTOP "Can't open registry
    Goto done
goon:
  System::Call "${RegQueryValueEx}(r0, '${VALUE}', 0,
  StrCmp $3 0 read
    MessageBox MB OK|MB_ICONSTOP "Can't query registry
    Goto done
read:
  StrCmp $1 ${REG MULTI SZ} multisz
    MessageBox MB OK|MB ICONSTOP "Registry value no RE
    Goto done
multisz:
  StrCmp $2 0 0 multiszalloc
    MessageBox MB OK|MB ICONSTOP "Registry value empty
```

```
Goto done
multiszalloc:
  System::Alloc $2
  Pop $1
  StrCmp $1 0 0 multiszget
    MessageBox MB OK|MB ICONSTOP "Can't allocate enoug
    Goto done
multiszget:
  System::Call "${RegQueryValueEx}(r0, '${VALUE}', 0,
  StrCmp $3 0 multiszprocess
    MessageBox MB_OK|MB_ICONSTOP "Can't query registry
    Goto done
multiszprocess:
  StrCpy $4 $1
  loop:
    System::Call "*$4(&t${NSIS_MAX STRLEN} .r3)"
    StrCmp $3 "" done
    DetailPrint $3
    StrLen $5 $3
    Int0p $4 $4 + $5
    Int0p $4 $4 + 1
    Goto loop
done:
  System::Free $1
```

StrCmp \$0 0 noClose
 System::Call "\${RegCloseKey}(r0)"

noClose:

SectionEnd

D.6 Predefined Macros for Unicode support

There are two macros that can help you write scripts that work for both Unicode and ANSI installers. To figure out if the script is being compiled to generate a Unicode installer, use !ifdef to check for \${NSIS_UNICODE}. To see what the size of a character is, use \${NSIS_CHAR_SIZE}. It will be 1 for ANSI and 2 for Unicode installers.

Appendix E: Useful Headers

- File Functions Header
 - Introduction
 - Locate
 - GetSize
 - DriveSpace
 - GetDrives
 - GetTime
 - GetFileAttributes
 - GetFileVersion
 - GetExeName
 - GetExePath
 - GetParameters
 - GetOptions
 - GetOptionsS
 - GetRoot
 - GetParent
 - GetFileName
 - GetBaseName
 - GetFileExt
 - BannerTrimPath
 - DirState
 - RefreshShellIcons
- Text Functions Header
 - Introduction
 - LineFind
 - LineRead
 - FileReadFromEnd
 - LineSum
 - FileJoin
 - TextCompare
 - TextCompareS
 - ConfigRead
 - ConfigReadS
 - ConfigWrite

- ConfigWriteS
- FileRecode
- TrimNewLines
- Word Functions Header
 - Introduction
 - WordFind
 - WordFindS
 - WordFind2X
 - WordFind2XS
 - WordFind3X
 - WordFind3XS
 - WordReplace
 - WordReplaceS
 - WordAdd
 - WordAddS
 - WordInsert
 - WordInsertS
 - StrFilter
 - StrFilterS
 - VersionCompare
 - VersionConvert

E.1 File Functions Header

; \$R0="C:\My Downloads"

E.1.1 Introduction

```
Include header:
```

E.1.2 Locate

SectionEnd

• Find files, directories and empty directories with mask and size options.

```
/L=FDE - Locate Files and Empty
                /M=[mask]
                    /M=*.*
                                    - Locate all (defau
                    /M=*.doc

    Locate Work.doc,

                    /M=Pho*
                                   - Locate PHOTOS, ph
                    /M=win???.exe - Locate winamp.exe
                    /M=winamp.exe
                                   - Locate winamp.exe
                /S=No:No[B|K|M|G]
                    /S=
                             - Don't locate file size
                             - Locate only files of 0
                    /S=0:0B
                    /S=5:9K
                             - Locate only files of 5
                             - Locate only files of 10
                    /S=:10M
                             - Locate only files of 1
                    /S=1G
                /G = [1|0]

    Locate with subdirector

                    /G=1
                             - Locate without subdirec
                    /G=0
                /B = [0|1]
                             - Banner isn't used (defa
                    /B=0
                    /B=1
                             - Banner is used. Callbac
                               start to search in new
              : Callback function when found
"Function"
Function "Function"
        ; $R9
                 "path\name"
        ; $R8
                 "path"
        ; $R7
                 "name"
                         ($R6="" if directory, $R6="0"
        ; $R6
                 "size"
        ; $R0-$R5 are not used (save data in them).
        ; . . .
        Push $var ; If $var="StopLocate" Then exit
FunctionEnd
```

Note: - Error flag if disk or directory isn't exist

- Error flag if syntax error

- See also: Locate plugin

Example (Find one file):

```
Section
    ${Locate} "C:\ftp" "/L=F /M=RPC DCOM.rar /S=1K
    ; 'RPC DCOM.rar' file in 'C:\ftp' with size 1

    IfErrors 0 +2
    MessageBox MB_OK "Error" IDOK +2
    MessageBox MB_OK "$$R0=$R0"

SectionEnd

Function Example1
    StrCpy $R0 $R9
    ; $R0="C:\ftp\files\RPC DCOM.rar"

    MessageBox MB_YESNO '$R0$\n$\nFind next?' IDYE
    StrCpy $0 StopLocate

    Push $0

FunctionEnd
```

Example (Write results to a text file):

```
GetTempFileName $R0
FileOpen $R1 $R0 w
${Locate} "C:\ftp" "/S=:2M /G=0" "Example2"
; folders and all files with size 2 Mb or less
; don't scan subdirectories
FileClose $R1

IfErrors 0 +2
MessageBox MB_OK "Error" IDOK +2
Exec '"notepad.exe" "$R0"'

SectionEnd

Function Example2
```

```
StrCmp $R6 '' 0 +3
FileWrite $R1 "Directory=$R9$\r$\n"
goto +2
FileWrite $R1 "File=$R9 Size=$R6 Mb$\r$\n"

Push $0
FunctionEnd
```

Example (Write results to an INI file):

```
Section

GetTempFileName $R0

${Locate} "C:\ftp" "/L=F /S=0K" "Example3"

; all files in 'C:\ftp' with size detect in Kb

IfErrors 0 +2

MessageBox MB_OK "Error" IDOK +2

Exec '"notepad.exe" "$R0"'

SectionEnd

Function Example3

WriteINIStr $R0 "$R8" "$R7" "$R6 Kb"

Push $0

FunctionEnd
```

Example (Delete empty directories):

```
StrCmp $R1 0 0 loop

IfErrors 0 +2
MessageBox MB_OK 'error' IDOK +2
MessageBox MB_OK '$R2 directories were removed
SectionEnd

Function Example4
MessageBox MB_YESNOCANCEL 'Delete empty "$R9"?
RMDir $R9
IntOp $R1 $R1 + 1
goto end

cancel:
StrCpy $R0 StopLocate

end:
Push $R0
FunctionEnd
```

Example (Move all files into one folder):

```
Function Example5
    StrCmp $R8 $R1 +6
    IfFileExists '$R1\$R7' +4
    Rename $R9 '$R1\$R7'
    IntOp $R2 $R2 + 1
    goto +2
    IntOp $R3 $R3 + 1

Push $0
FunctionEnd
```

Example (Copy files with log):

```
Section
        StrCpy $R0 "C:\ftp" ;Directory copy from
StrCpy $R1 "C:\ftp2" ;Directory copy into
        StrLen $R2 $R0
        GetTempFileName $0
        FileOpen $R3 $0 w
         ${Locate} "$R0" "/L=FDE" "Example6"
         FileClose $R3
         IfErrors 0 +2
        MessageBox MB OK 'error'
         Exec '"notepad.exe" "$0"' ;view log
SectionEnd
Function Example6
        StrCpy $1 $R8 '' $R2
        StrCmp $R6 '' 0 +3
        CreateDirectory '$R1$1\$R7'
        goto end
        CreateDirectory '$R1$1'
        CopyFiles /SILENT $R9 '$R1$1'
```

Example (Recreate directory structure):

```
Section

StrCpy $R0 "C:\ftp" ;Directory structure f
StrCpy $R1 "C:\ftp2" ;Directory structure i
StrLen $R2 $R0

${Locate} "$R0" "/L=D" "Example7"

IfErrors 0 +2
MessageBox MB_OK 'error'

SectionEnd

Function Example7
StrCpy $1 $R9 '' $R2
CreateDirectory '$R1$1'

Push $0

FunctionEnd
```

Example (Locate with banner - NxS plugin required):

E.1.3 GetSize

- Find the size of a file, files mask or directory.
- Find the sum of the files, directories and subdirectories.

```
/S=No:No[B|K|M|G]
                     /S= - Don't find file size (f
/S=0:0B - Find only files of 0 By
                     /S=5:9K - Find only files of 5 to
                     /S=:10M - Find only files of 10 M
                               - Find only files of 1 Gi
                     /S=1G
                /G=[1|0]
                              - Find with subdirectorie
                     /G=1
                     /G=0 - Find without subdirecto
               ; Result1: Size
$var1
               ; Result2: Sum of files
$var2
                 Result3: Sum of directories
$var3
```

Note:

- Error flag if disk or directory isn't exist
- Error flag if syntax error
- See also: Locate plugin

Examples:

```
Section 'Find file size of "$WINDIR\Explorer.exe" in K

${GetSize} "$WINDIR" "/M=Explorer.exe /S=0K /G
; $0="220" KiB
; $1="1" files
; $2="" directories

IfErrors 0 +2
MessageBox MB_OK "Error"
SectionEnd
```

```
Section 'Find folder size of "C:\Installs\Drivers" in

${GetSize} "C:\Installs\Drivers" "/S=0M" $0 $1
; $0="132" MiB
; $1="555" files
; $2="55" directories
```

```
IfErrors 0 +2
   MessageBox MB_OK "Error"
SectionEnd

Section 'Find sum of files and folders in "$WINDIR" (n
   ${GetSize} "$WINDIR" "/G=0" $0 $1 $2
   ; $0="" size
   ; $1="253" files
   ; $2="46" directories

IfErrors 0 +2
   MessageBox MB_OK "Error"
SectionEnd
```

E.1.4 DriveSpace

• Get total, occupied or free space of the drive.

Note:

- Error flag if disk isn't exist or not ready
- Error flag if syntax error

Example:

E.1.5 GetDrives

• Find all available drives in the system.

```
${GetDrives} "[Option]" "Function"
```

```
"[Option]"
               ; [FDD+HDD+CDROM+NET+RAM]
                   FDD
                         Floppy Disk Drives
                         Hard Disk Drives
                   HDD
                  CDROM CD-ROM Drives
                  NET
                         Network Drives
                  RAM RAM Disk Drives
                 [ALL]
                   Find all drives by letter (default
"Function" : Callback function when found
Function "Function"
       ; $9  "drive letter" (a:\ c:\ ...)
       ; $8 "drive type" (FDD HDD ...)
       ; $R0-$R9 are not used (save data in them).
       ; ...
```

```
Push $var ; If $var="StopGetDrives" Then ex
FunctionEnd
```

Example1:

```
Section
${GetDrives} "FDD+CDROM" "Example1"
SectionEnd

Function Example1
MessageBox MB_OK "$9 ($8 Drive)"

Push $0
FunctionEnd
```

Example2:

```
Section
${GetDrives} "ALL" "Example2"
SectionEnd

Function Example2
MessageBox MB_OK "$9 ($8 Drive)"

Push $0
FunctionEnd
```

Example3 (Get type of drive):

```
Section
StrCpy $R0 "D:\" ;Drive letter
StrCpy $R1 "invalid"

${GetDrives} "ALL" "Example3"

MessageBox MB_OK "Type of drive $R0 is $R1"
SectionEnd
```

```
Function Example3
StrCmp $9 $R0 0 +3
StrCpy $R1 $8
StrCpy $0 StopGetDrives

Push $0
FunctionEnd
```

E.1.6 GetTime

- Get local or system time.
- Get file time (access, creation and modification).

```
${GetTime} "[File]" "[Option]" $var1 $var2 $var3 $var4
```

```
Ignored if "L" or "LS"
"[File]"
"[Option]"
                  [Options]
                   L Local time
                       last Access file time
                      Creation file time
                    C
                   M Modification file time
                   LS System time (UTC)
                    AS last Access file time (UTC)
                    CS Creation file time (UTC)
                   MS Modification file time (UTC)
                ; Result1: day
$var1
                ; Result2: month
$var2
                ; Result3: year
$var3
$var4
                ; Result4: day of week name
                ; Result5: hour
$var5
                ; Result6: minute
$var6
$var7
                 Result7: seconds
```

Note:

- Error flag if file isn't exist
- Error flag if syntax error
- See also: Time plugin

Examples:

```
Section 'Get local time'
        ${GetTime} "" "L" $0 $1 $2 $3 $4 $5 $6
        ; $0="01"
                       day
        ; $1="04"
                       month
        ; $2="2005"
                       year
        ; $3="Friday" day of week name
        ; $4="16"
                       hour
        ; $5="05"
                      minute
        ; $6="50" seconds
        MessageBox MB OK 'Date=$0/$1/$2 ($3)$\nTime=$4
SectionEnd
Section 'Get file time'
        ${GetTime} "$WINDIR\Explorer.exe" "C" $0 $1 $2
        ; $0="12"
                        day
        ; $1="10"
                        month
        ; $2="2004"
                        year
        ; $3="Tuesday"
                        day of week name
        ; $4="2"
                        hour
        ; $5="32"
                        minute
        ; $6="03"
                        seconds
        IfErrors 0 +2
        MessageBox MB OK "Error" IDOK +2
        MessageBox MB OK 'Date=$0/$1/$2 ($3)$\nTime=$4
SectionEnd
Section 'Get system time'
        ${GetTime} "" "LS" $0 $1 $2 $3 $4 $5 $6
```

```
; $0="01"
                       day
        ; $1="04"
                       month
        ; $2="2005"
                       year
        ; $3="Friday"
                       day of week name
        ; $4="11"
                       hour
        ; $5="05"
                       minute
        ; $6="50" seconds
        MessageBox MB OK 'Date=$0/$1/$2 ($3)$\nTime=$4
SectionEnd
Section 'Convert time to 12-hour format AM/PM'
        ${GetTime} "" "L" $0 $1 $2 $3 $4 $5 $6
        StrCmp $4 0 0 +3
        StrCpy $4 12
        qoto +3
        StrCmp $4 12 +5
        IntCmp $4 12 0 0 +3
        StrCpy $7 AM
        qoto +3
        IntOp $4 $4 - 12
        StrCpy $7 PM
        MessageBox MB OK 'Date=$0/$1/$2 ($3)$\nTime=$4
SectionEnd
```

E.1.7 GetFileAttributes

• Get attributes of file or directory.

```
${GetFileAttributes} "[File]" "[Attributes]" $var

"[File]" ; File or directory
;
```

Note:

- Error flag is set if file doesn't exist

Example:

E.1.8 GetFileVersion

• Get version information from executable file.

```
${GetFileVersion} "[Executable]" $var
"[Executable]" ; Executable file (*.exe *.dll ...
```

\$var ; Result: Version number

Note:

- Error flag if file doesn't exist
- Error flag if file doesn't contain version information

Example:

```
${GetFileVersion} "C:\ftp\program.exe" $R0; $R0="1.1.
```

E.1.9 GetExeName

• Get installer filename (with valid case for Windows 98/Me).

Syntax:

```
${GetExeName} $var
```

Example:

```
${GetExeName} $R0 ; $R0="C:\ftp\program.exe"
```

E.1.10 GetExePath

• Get installer pathname (\$EXEDIR with valid case for Windows 98/Me).

Syntax:

```
${GetExePath} $var
```

Example:

```
\{GetExePath\} \ R0 \ ; \ R0="C:\ftp"
```

E.1.11 GetParameters

• Get command line parameters.

Syntax:

```
${GetParameters} $var
```

Example:

```
${GetParameters} $R0 ; $R0="[parameters]"
```

E.1.12 GetOptions

• Get options from command line parameters.

Syntax:

```
${GetOptions} "[Parameters]" "[Option]" $var

"[Parameters]" ; command line parameters
;

"[Option]" ; option name
;

$var ; Result: option string
```

Note:

- The error flag is set if the option is not found
- The first character in the option string is treated as a parameter delimiter

Example1:

```
Section
${GetOptions} "/S /T" "/T" $R0

IfErrors 0 +2

MessageBox MB_OK "Not found" IDOK +2

MessageBox MB_OK "Found"

SectionEnd
```

Example2:

```
Section
    ${GetOptions} "-INSTDIR=C:\Program Files\Commo
    ;$R0=C:\Program Files\Common Files
SectionEnd
```

Example3:

Example4:

```
Section
     ${GetOptions} `-SILENT=yes -INSTDIR='"C:/Progr
;$R0="C:/Program Files/Common Files"
SectionEnd
```

E.1.13 GetOptionsS

• Same as GetOptions, but case sensitive.

E.1.14 GetRoot

Get root directory.

Syntax:

```
${GetRoot} "[FullPath]" $var
```

Examples:

```
${GetRoot} "C:\Program Files\NSIS" $R0 ; $R0="C:"
${GetRoot} "\\SuperPimp\NSIS\Source\exehead\Ui.c" $R0
```

E.1.15 GetParent

• Get parent directory.

Syntax:

```
${GetParent} "[PathString]" $var
```

Example:

\${GetParent} "C:\Program Files\Winamp\uninstwa.exe" \$R

E.1.16 GetFileName

• Get last part from directory path.

Syntax:

```
${GetFileName} "[PathString]" $var
```

Example:

\${GetFileName} "C:\Program Files\Winamp\uninstwa.exe"

E.1.17 GetBaseName

Get file name without extension.

Syntax:

```
${GetBaseName} "[FileString]" $var
```

Example:

\${GetBaseName} "C:\ftp\program.exe" \$R0 ; \$R0="program"

E.1.18 GetFileExt

Get extension of file.

```
${GetFileExt} "[FileString]" $var
```

Example:

```
${GetFileExt} "C:\ftp\program.exe" $R0 ; $R0="exe"
```

E.1.19 BannerTrimPath

• Trim string path for banner.

Syntax:

```
${BannerTrimPath} "[PathString]" "[Option]" $var
```

```
"[PathString]"
"[Option]"
                    [Length][A|B|C|D]
                    Length
                             -Maximum string length
                             -Trim center path (default
                      Α
                               (C:\root\...\third path)
                               If A mode not possible T
                      В
                             -Trim right path
                               (C:\root\second path\...
                               If B mode not possible T
                             -Trim right string
                      C
                               (C:\root\second path\thi
                      D
                             -Trim right string + filen
                               (C:\root\second p...\thi
                               If D mode not possible T
                    Result:
                             Trimmed path
$var
```

Example:

```
Section
    ${BannerTrimPath} "C:\Server\Documents\Termina
;$R0=C:\Server\...\Terminal\license.htm
```

Example (Banner plugin):

```
!include "WinMessages.nsh"
!include "FileFunc.nsh"
Section
        Banner::show "Starting..."
        Banner::getWindow
        Pop $R1
        ${Locate} "$WINDIR" "/L=F /M=*.* /B=1" "Locate
        Banner::destroy
SectionEnd
Function LocateCallback
        StrCmp $R0 $R8 code
        StrCpy $R0 $R8
        ${BannerTrimPath} "$R8" "38B" $R8
        GetDlgItem $1 $R1 1030
        SendMessage $1 ${WM SETTEXT} 0 "STR:$R8"
        code:
        StrCmp $R9 '' end
        ; . . .
        end:
        Push $0
FunctionEnd
```

Example (NxS plugin):

```
${Locate} "$WINDIR" "/L=F /M=*.* /B=1" "Locate
        nxs::Destroy
SectionEnd
Function LocateCallback
        StrCmp $R0 $R8 abortcheck
        StrCpy $R0 $R8
        ${BannerTrimPath} "$R8" "55A" $R8
        nxs::Update /NOUNLOAD /sub "$R8" /pos 78 /end
        abortcheck:
        nxs::HasUserAborted /NOUNLOAD
        Pop $0
        StrCmp $0 1 0 +2
        StrCpy $0 StopLocate
        StrCmp $R9 '' end
        ; . . .
        end:
        Push $0
FunctionEnd
```

E.1.20 DirState

• Check directory full, empty or not exist.

Example:

```
${DirState} "$TEMP" $R0 ; $R0="1" (directory is full)
```

E.1.21 RefreshShellIcons

• After changing file associations, you can call this function to refresh the shell immediately.

Syntax:

```
${RefreshShellIcons}
```

Example:

```
Section
```

```
WriteRegStr HKCR "Winamp.File\DefaultIcon" ""
${RefreshShellIcons}
```

SectionEnd

Previous | Contents | Next

Previous | Contents | Next

E.2 Text Functions Header

E.2.1 Introduction

Include header:

```
!include "TextFunc.nsh"
```

Call functions:

```
Section Install
    ${LineRead} "C:\a.log" "-1" $R0
    ; $R0="Last line$\r$\n"
SectionEnd
```

E.2.2 LineFind

• Find specified lines in text file, and edit or view these lines in callback function.

```
"[LineNumbers]"
                    [No|-No|No:No|{No}|{-No}|{No:No}]
                               all lines to change (de
                      1:-1
                               second line from start
                      2
                      - 3
                               third line from end
                      5:9
                               range of lines from 5 t
                      {2}
                               only second line from s
                      {-3}
                               only third line from en
                               only range of lines fro
                      {5:9}
"Function"
                  ; Callback function for specified li
Function "Function"
          $R9
                    current line
        ; $R8
                    current line number
        ; $R7
                    current line negative number
        ; $R6
                    current range of lines
        ; $R5
                    handle of a file opened to read
        ; $R4
                    handle of a file opened to write (
        ; you can use any string functions
        ; $R0-$R3 are not used (save data in them).
        ; ...
        Push $var
                      ; If $var="StopLineFind"
                                                 Then
                       ; If $var="SkipWrite"
                                                 Then
FunctionEnd
```

Note: - Error flag if input file doesn't exist

- Error flag if output file path doesn't exist
- Ranges must be specified on growth (2 4:5 9:-8 -5:-4 -2:-1)
- Output file will not be updated if no changes made.

Example1 (delete first two symbols):

```
Section
${LineFind} "C:\a.log" "C:\a-edited.log" "3:-1
```

```
IfErrors 0 +2
    MessageBox MB_OK "Error"
SectionEnd

Function Example1
    ${TrimNewLines} '$R9' $R9
    StrCpy $R9 $R9 '' 2
    StrCpy $R9 '$R9$\r$\n'
    ;start from 3 line and delete first two symbol
    Push $0
FunctionEnd
```

Example2 (show changed lines):

Example3 (delete lines):

```
Push $0
FunctionEnd
```

Example4 (insert lines):

Example5 (replace in file with count of changes - "WordFunc.nsh" required):

```
IntOp $R0 $R0 + 1
;$R0 count of changed lines

Push $0
FunctionEnd
```

Example6 (line string to cut or delete):

```
Section
        ${LineFind} "\a.log" "C:\logs\a.log" "" "Examp
        IfErrors 0 +2
        MessageBox MB OK "Error" IDOK +2
        MessageBox MB OK "Processed lines=$R1:$R2"
SectionEnd
Function Example6
        ;(Cut lines from a line to another line (also
        StrCmp $R0 finish stop
        StrCmp $R0 start finish
        StrCmp $R9 'Start Line$\r$\n' 0 skip
        StrCpy $R0 start
        StrCpy $R1 $R8
        goto code
        finish:
        StrCmp $R9 'Finish Line$\r$\n' 0 code
        StrCpy $R0 finish
        StrCpy $R2 $R8
        goto code
        skip:
        StrCpy $0 SkipWrite
        goto output
        stop:
        StrCpy $0 StopLineFind
        goto output
        ;;(Delete lines from a line to another line (a
        ; StrCmp $R0 finish code
```

```
; StrCmp $R0 start finish
        ; StrCmp $R9 'Start Line$\r$\n' 0 code
        ; StrCpy $R0 start
        ; StrCpy $R1 $R8
        ; goto skip
        ; finish:
        ; StrCmp $R9 'Finish Line$\r$\n' 0 skip
        ; StrCpy $R0 finish
        ; StrCpy $R2 $R8
        ; skip:
        ; StrCpy $0 SkipWrite
        ; goto output
        code:
        ; . . .
        output:
        Push $0
FunctionEnd
```

Example7 (read lines):

E.2.3 LineRead

• Get line in file specified with number.

Syntax:

Note:

- Error flag if input file doesn't exist
- Error flag if line number not found

Example:

E.2.4 FileReadFromEnd

• Read text file from end line by line.

Note:

- Error flag if input file doesn't exist

Example1:

```
Section
${FileReadFromEnd} "C:\a.log" "Example1"

IfErrors 0 +2
MessageBox MB_OK "Error"

SectionEnd

Function Example1
MessageBox MB_OKCANCEL '"Line"=[$9]$\n "#"=[
StrCpy $0 StopFileReadFromEnd

Push $0

FunctionEnd
```

Example2 (Reverse text file):

```
Section
GetTempFileName $R0
FileOpen $R1 $R0 w
${FileReadFromEnd} "C:\a.log" "Example2"
FileClose $R1

IfErrors 0 +2
MessageBox MB_OK "Error" IDOK +2
```

E.2.5 LineSum

• Get sum of lines in text file.

Syntax:

```
${LineSum} "[File]" $var

"[File]" ; Input file
$var ; Result: Sum of lines
```

Note:

- Error flag if input file doesn't exist

Example:

E.2.6 FileJoin

• Join two files in one (File1 + File2 = File3).

Syntax:

```
${FileJoin} "[File1]" "[File2]" "[File3]"

"[File1]" ; Input File1

"[File2]" ; Input File2

"[File3]" ; Output File3
 ; If [File3]="" Then add [File2] to [File3]
```

Note:

- Error flag if input files don't exist
- Error flag if output file path doesn't exist

Example1 (Join: a.log + b.log = Z.log):

Example2 (Add: a.log + b.log = a.log):

E.2.7 TextCompare

• Compare two text files.

```
FastDiff
                          Compare line N (File1) with
                          Call function if Different
              FastEqual
                          Compare line N (File1) with
                          Call function if Equal line
               (line number independent):
               SlowDiff
                          Compare line N (File1) with
                          Call function if line N (Fi
                          Compare line N (File1) with
               SlowEqual
                          Call function if line N (Fi
"Function"
             ; Callback function
Function "Function"
               "Line File1"
       ; $9
       ; $8
               "Line number"
       ; $7
               "Line File2" (empty if SlowDiff)
       ; $6
               "Line number" (empty if SlowDiff)
       ; $RO-$R9 are not used (save data in them).
       Push $var ; If $var="StopTextCompare"
                                                 Then
FunctionEnd
```

Note:

- Error flag if File1 or File2 doesn't exist
- Error flag if syntax error

Example (Different or Equal):

```
Section
StrCpy $R0 ''
${TextCompare} "C:\1.txt" "C:\2.txt" "FastDiff
IfErrors 0 +2
MessageBox MB_OK "Error" IDOK +4

StrCmp $R0 NotEqual 0 +2
MessageBox MB_OK "Files differ" IDOK +2
```

```
MessageBox MB_OK "Files identical"
SectionEnd

Function Example1
StrCpy $R0 NotEqual
StrCpy $0 StopTextCompare

Push $0

FunctionEnd
```

Example (Compare line-by-line - Different):

```
Section
        StrCpy $R0 'Text1.txt'
        StrCpy $R1 'Text2.txt'
        GetTempFileName $R2
        FileOpen $R3 $R2 w
        FileWrite $R3 "$R0 | $R1$\r$\n"
        ${TextCompare} "$R0" "$R1" "FastDiff" "Example
        IfErrors 0 +2
        MessageBox MB OK "Error" IDOK +2
        Exec "notepad.exe $R2"
FunctionEnd
Function Example2
        FileWrite $R3 '$8=$9'
        FileWrite R3 '$6=$7$\r$\n'
        Push $0
FunctionEnd
```

Example (Compare line-by-line - Equal):

```
Section
StrCpy $R0 'Text1.txt'
StrCpy $R1 'Text2.txt'
```

```
GetTempFileName $R2
    FileOpen $R3 $R2 w
    FileWrite $R3 "$R0 | $R1$\r$\n"
    ${TextCompare} "$R0" "$R1" "FastEqual" "Exampl
    IfErrors 0 +2
    MessageBox MB_OK "Error" IDOK +2

    Exec "notepad.exe $R2"
FunctionEnd

Function Example3
    FileWrite $R3 '$8|$6=$9'

    Push $0
FunctionEnd
```

Example (Compare all lines - Different):

```
StrCpy $R0 'Text1.txt'
StrCpy $R1 'Text2.txt'

GetTempFileName $R2
FileOpen $R3 $R2 w
FileWrite $R3 "$R0 | $R1$\r$\n"
${TextCompare} "$R0" "$R1" "SlowDiff" "Example
IfErrors 0 +2
MessageBox MB_OK "Error" IDOK end

FileWrite $R3 "$\r$\n$R1 | $R0$\r$\n"
${TextCompare} "$R1" "$R0" "SlowDiff" "Example
FileClose $R3
IfErrors 0 +2
MessageBox MB_OK "Error" IDOK end

Exec "notepad.exe $R2"
```

```
end:
FunctionEnd

Function Example4
FileWrite $R3 '$8=$9'

Push $0
FunctionEnd
```

Example (Compare all lines - Equal):

```
Section

StrCpy $R0 'Text1.txt'
StrCpy $R1 'Text2.txt'

GetTempFileName $R2
FileOpen $R3 $R2 w
FileWrite $R3 "$R0 | $R1$\r$\n"
${TextCompare} "$R0" "$R1" "SlowEqual" "Exampl
IfErrors 0 +2
MessageBox MB_OK "Error" IDOK +2

Exec "notepad.exe $R2"

FunctionEnd

Function Example5
FileWrite $R3 '$8|$6=$9'

Push $0

FunctionEnd
```

Example (Show variables):

```
Section
    ${TextCompare} "C:\1.txt" "C:\2.txt" "FastDiff
    IfErrors 0 +2
```

E.2.8 TextCompareS

• Same as TextCompare, but case sensitive.

E.2.9 ConfigRead

• Read value from entry name in config file.

Syntax:

Note:

- Error flag if entry not found
- Error flag if file doesn't exist

Example1:

```
Section
    ${ConfigRead} "C:\AUTOEXEC.BAT" "SET winbootdi
;$R0=C:\WINDOWS
SectionEnd
```

Example2:

E.2.10 ConfigReadS

Same as ConfigRead, but case sensitive.

E.2.11 ConfigWrite

• Write value from entry name in config file.

Syntax:

```
${ConfigWrite} "[File]" "[Entry]" "[Value]" $var
"[File]"
              ; config file
"[Entry]"
              ; entry name
"[Value]"
              ; value name
                 if "" then delete Entry
                Result:
$var
                   $var=CHANGED Value is written
                                 Entry is deleted
                   $var=DELETED
                                 Entry and Value are a
                   $var=ADDED
                                 Entry and Value alrea
                   $var=SAME
```

Note:

- Error flag if file doesn't exist
- Error flag if file can't be opened

Example1:

Example2:

Example3:

```
Section
          ${ConfigWrite} "C:\apache\conf\httpd.conf" "Ti
          ;$R0=DELETED
SectionEnd
```

E.2.12 ConfigWriteS

• Same as ConfigWrite, but case sensitive.

E.2.13 FileRecode

• Recode text file from DOS to Windows format and vice-versa.

```
${FileRecode} "[File]" "[Format]"
"[File]" ;
;
```

```
"[Format]" ; OemToChar -from DOS to Windows
; CharToOem -from Windows to DOS
```

Note:

- Error flag if file doesn't exist
- Error flag if syntax error

Example:

E.2.14 TrimNewLines

• Trim newlines in a string.

Syntax:

```
${TrimNewLines} "[string]" $var

"[string]" ; Input string
$var ; Result: String without '$\r' and '$\n'
Example:
```

Previous | Contents | Next

Previous | Contents | Next

E.3 Word Functions Header

E.3.1 Introduction

Include header:

```
!include "WordFunc.nsh"
```

Call functions:

```
Section Install
${WordFind} "A--H---S" "-" "+2" $R0
; $R0="H"
SectionEnd
```

```
Section un.Install
     ${WordReplace} "A--H---S" "-" "x" "+3*" $R0
     ; $R0="A--HxS"
SectionEnd
```

E.3.2 WordFind

• Multi-features string function.

```
Strings:
"[word+1][delimiter][word+2][delimiter][word+3]..."
"[delimiter][word+1][delimiter][word+2][delimiter]..."
"[delimiter][delimiter][word+1][delimiter][delimiter][
"...[word-3][delimiter][word-2][delimiter][word-1]"
"...[delimiter][word-2][delimiter][delimiter]"
```

```
${WordFind} "[string]" "[delimiter]" "[E][options]" $v
```

```
"[string]"
                   ;[string]
                      input string
"[delimiter]"
                    :[delimiter]
                      one or several symbols
"[E][options]"
                    ; [options]
                                 : word number from sta
                      +number
                                 : word number from end
                      -number
                                 : delimiter number fro
                      +number}
                                   all space after this
                                   delimiter to output
                                 : delimiter number fro
                      +number{
                                   all space before thi
                                   delimiter to output
                      +number}} : word number from sta
                                   all space after this
                                   to output
                      +number{{ : word number from sta
                                   all space before thi
                                   to output
                      +number{} : word number from sta
                                   all space before and
                                   this word (word excl
                      +number*} : word number from sta
                                   all space after this
                                   word to output with
                      +number{* : word number from sta
                                   all space before thi
                                   word to output with
                                 : sum of words to outp
                      #
                      *
                                 : sum of delimiters to
                                 : number of word to ou
                      /word
                   ;[E]
                      with errorlevel output
                      IfErrors:
                                 delimiter not found
                         $var=1
                                  no such word number
                          $var=2
```

```
; $var=3 syntax error (Use: +1
;[]
; no errorlevel output (default)
; If some errors found then (resul
;
$var ;output (result)
```

Note: - Accepted numbers 1,01,001,...

Example (Find word by number):

```
Section
     $\{\WordFind\}\ "C:\io.sys C:\Program Files C:\\WIN
     ; \$\R0=\Program Files\"
SectionEnd
```

Example (Delimiter exclude):

Example (Sum of words):

Example (Sum of delimiters):

Example (Find word number):

```
Section
     $\{\text{WordFind}\} "C:\io.sys C:\Program Files C:\\WIN
     ; \$\{\text{R0}="3"}\]
SectionEnd
```

Example (}}):

Example ({}):

Example (*):

Example (Get parent directory):

Example (Coordinates):

Example (With errorlevel output):

```
Section

${\text{WordFind}}$ "[string]" "[delimiter]" "E[option

IfErrors 0 end

StrCmp $R0 1 0 +2 ; errorlevel 1?

MessageBox MB_OK 'delimiter not found' IDOK en

StrCmp $R0 2 0 +2 ; errorlevel 2?

MessageBox MB_OK 'no such word number' IDOK en

StrCmp $R0 3 0 +2 ; errorlevel 3?

MessageBox MB_OK 'syntax error'

end:

SectionEnd
```

Example (Without errorlevel output):

Example (If found):

```
Section

${WordFind} "C:\io.sys C:\logo.sys" ":\lo" "E+

IfErrors notfound found
found:
    MessageBox MB_OK 'Found' IDOK end
    notfound:
    MessageBox MB_OK 'Not found'

end:

SectionEnd
```

Example (If found 2):

```
Section

${WordFind} "C:\io.sys C:\logo.sys" ":\lo" "+1

StrCmp $R0 "C:\io.sys C:\logo.sys" notfound fo found:

MessageBox MB_OK 'Found' IDOK end notfound:

MessageBox MB_OK 'Not found'

end:

SectionEnd
```

Example (To accept one word in string if delimiter not found):

```
Section
StrCpy $0 'OneWord'
StrCpy $1 1

loop:
${WordFind} "$0" " "E+$1" $R0
IfErrors 0 code
StrCmp $1$R0 11 0 error
```

```
StrCpy $R0 $0
    goto end

code:
    ; ...
    IntOp $1 $1 + 1
    goto loop

    error:
    StrCpy $1 ''
    StrCpy $R0 ''

    end:
    ; $R0="OneWord"

SectionEnd
```

E.3.3 WordFindS

• Same as WordFind, but case sensitive.

E.3.4 WordFind2X

• Find word between two delimiters.

```
Strings:
"[delimiter1][word+1][delimiter2][delimiter1][word+2][
"[text][delimiter1][text][delimiter1][word+1][delimite
"...[delimiter1][word-2][delimiter2][delimiter1][word-
"...[text][delimiter1][text][delimiter1][word-1][delim
```

```
first delimiter
"[delimiter2]"
                   ;[delimiter2]
                      second delimiter
"[E][options]"
                    :[options]
                      +number : word number from sta
                      -number : word number from end
                      +number}} : word number from sta
                                   after this word to o
                      +number{{ : word number from end
                                   before this word to
                      +number{} : word number from sta
                                   all space before and
                                   this word (word excl
                      +number*} : word number from sta
                                   all space after this
                                   word to output with
                      +number{* : word number from sta
                                   all space before thi
                                   word to output with
                                 : sum of words to outp
                      #
                      /word
                                 : number of word to ou
                    ;[E]
                      with errorlevel output
                      IfErrors:
                         $var=1 no words found
                         $var=2 no such word number
                                 syntax error (Use: +1
                         $var=3
                    ;[]
                      no errorlevel output (default)
                      If some errors found then (resul
                   ;output (result)
$var
```

Example (1):

Section

```
${WordFind2X} "[C:\io.sys];[C:\logo.sys];[C:\W
; $R0="logo.sys"
SectionEnd
```

Example (2):

Example (3):

```
Section
          ${WordFind2X} "C:\WINDOWS C:\io.sys C:\logo.sy
          ; $R0="C:\WINDOWS C:\io.sys C:"
SectionEnd
```

Example (4):

Example (5):

Example (6):

Example (With errorlevel output):

E.3.5 WordFind2XS

• Same as WordFind2X, but case sensitive.

E.3.6 WordFind3X

• Find a word that contains a string, between two delimiters.

```
-number : word number from end
                      +number}} : word number from sta
                                  after this word to o
                      +number{{ : word number from end
                                  before this word to
                      +number{} : word number from sta
                                  all space before and
                                  this word (word excl
                      +number*} : word number from sta
                                  all space after this
                                  word to output with
                      +number{* : word number from sta
                                  all space before thi
                                  word to output with
                                : sum of words to outp
                      #
                      /word
                                : number of word to ou
                   ;[E]
                      with errorlevel output
                      IfErrors:
                         $var=1 no words found
                         $var=2 no such word number
                         $var=3
                                 syntax error (Use: +1
                   ;[]
                      no errorlevel output (default)
                      If some errors found then (resul
                   ;output (result)
$var
```

Example (1):

Example (2):

Example (3):

Example (4):

Example (5):

Example (6):

Example (With errorlevel output):

```
IfErrors 0 noerrors
MessageBox MB_OK 'Errorlevel=$R0' IDOK end
noerrors:
    MessageBox MB_OK 'No errors'
    end:
SectionEnd
```

E.3.7 WordFind3XS

Same as WordFind3X, but case sensitive.

E.3.8 WordReplace

• Replace or delete word from string.

```
${WordReplace} "[string]" "[word1]" "[word2]" "[E][opt
"[string]"
                   ;[string]
                      input string
"[word1]"
                   ; [word1]
                     word to replace or delete
"[word2]"
                   ; [word2]
                      replace with (if empty delete)
"[E][options]"
                   ; [options]
                      +number : word number from star
                     -number : word number from end
                     +number* : word number from star
                     -number* : word number from end
                               : replace all results
                     +*
                               : multiple-replace all
                               : if exists replace all
                                   from left edge
```

```
}
                                : if exists replace all
                                    from right edge
                                : if exists replace all
                      {}
                                    from edges
                                : if exists multiple-re
                      {*
                                    delimiters from lef
                      }*
                                : if exists multiple-re
                                    delimiters from rig
                      {}*
                                : if exists multiple-re
                                    delimiters from edg
                    ;[E]
                      with errorlevel output
                      IfErrors:
                         $var=1 word to replace not f
                         $var=2 no such word number
                         $var=3
                                  syntax error (Use: +1
                    ;[]
                      no errorlevel output (default)
                      If some errors found then (resul
                   ;output (result)
$var
```

Example (replace):

Example (delete):

```
Section
     ${WordReplace} "C:\io.sys C:\logo.sys C:\WINDO
     ; $R0="C:\io. C:\logo. C:\WINDOWS"
SectionEnd
```

Example (multiple-replace 1):

Example (multiple-replace 2):

```
Section
     ${WordReplace} "C:\io.sys C:\logo.sysSYSsys C:
     ; $R0="C:\io.bmp C:\logo.bmp C:\WINDOWS"
SectionEnd
```

Example (multiple-replace 3):

Example (With errorlevel output):

```
Section
    ${WordReplace} "C:\io.sys C:\logo.sys" "sys" "
    ; $R0="2" (no such word number "+3")

    IfErrors 0 noerrors
    MessageBox MB_OK 'Errorlevel=$R0' IDOK end

    noerrors:
    MessageBox MB_OK 'No errors'

    end:
SectionEnd
```

E.3.9 WordReplaceS

• Same as WordReplace, but case sensitive.

E.3.10 WordAdd

• Add words to string1 from string2 if not exist or delete words if exist.

Syntax:

```
${WordAdd} "[string1]" "[delimiter]" "[E][options]" $v
"[string1]"
                     ;[string1]
                        string for addition or removin
"[delimiter]"
                     :[delimiter]
                        one or several symbols
"[E][options]"
                     :[options]
                        +string2 : words to add
                        -string2 : words to delete
                     ;[E]
                        with errorlevel output
                        IfErrors:
                           $var=1 delimiter is empty
                           $var=3
                                   syntax error (use:
                     ;[]
                        no errorlevel output (default)
                        If some errors found then (res
                     ;output (result)
$var
```

Example (add):

Example (delete):

```
Section
     ${WordAdd} "C:\io.sys C:\logo.sys C:\WINDOWS"
```

```
; $R0="C:\logo.sys"
SectionEnd
```

Example (add to one):

```
Section
     $\{\text{WordAdd}\} \ \"C:\io.sys\" \ \" \ \"+C:\\WINDOWS C:\con
     ; \$\{\text{R0}=\"C:\io.sys C:\\WINDOWS C:\config.sys\"}
SectionEnd
```

Example (delete one):

Example (No new words found):

Example (No words deleted):

Example (With errorlevel output):

```
Section
    ${WordAdd} "C:\io.sys C:\logo.sys" "" "E-C:\lo
    ; $R0="1" (delimiter is empty "")
```

```
IfErrors 0 noerrors
MessageBox MB_OK 'Errorlevel=$R0' IDOK end
noerrors:
    MessageBox MB_OK 'No errors'
    end:
SectionEnd
```

E.3.11 WordAddS

• Same as WordAdd, but case sensitive.

E.3.12 WordInsert

Insert word in string.

Syntax:

```
${WordInsert} "[string]" "[delimiter]" "[word]" "[E][o
"[string]"
                    ;[string]
                       input string
"[delimiter]"
                    :[delimiter]
                       one or several symbols
"[word]"
                    ; [word]
                       word to insert
"[E][options]"
                    ; [options]
                       +number : word number from sta
                       -number : word number from end
                    :[E]
                       with errorlevel output
                       IfErrors:
                          $var=1 delimiter is empty
                          $var=2 wrong word number
```

```
; $var=3 syntax error (Use: +
;[]
; no errorlevel output (default)
; If some errors found then (resu
;
$var
;output (result)
```

Example (1):

Example (2):

Example (3):

Example (With errorlevel output):

```
Section
    ${WordInsert} "C:\io.sys C:\logo.sys" " " "C:\
    ; $R0="2" (wrong word number "+4")

    IfErrors 0 noerrors
    MessageBox MB_OK 'Errorlevel=$R0' IDOK end
    noerrors:
```

```
MessageBox MB_OK 'No errors'

end:
SectionEnd
```

E.3.13 WordInsertS

Same as WordInsert, but case sensitive.

E.3.14 StrFilter

- Convert string to uppercase or lowercase.
- Set symbol filter.

Syntax:

```
${StrFilter} "[string]" "[options]" "[symbols1]" "[sym
"[string]" ;[string]
```

```
input string
"[options]"
                 ;[+|-][1|2|3|12|23|31][eng|rus]
                    + : convert string to uppercase
                       : convert string to lowercase
                   1 : only Digits
                 ; 2 : only Letters
; 3 : only Special
                   12 : only Digits + Letters
                    23 : only Letters + Special
                    31 : only Special + Digits
                    eng : English symbols (default)
                    rus : Russian symbols
"[symbols1]"
                 ;[symbols1]
                    symbols include (not changeable)
"[symbols2]"
                 ;[symbols2]
```

```
; symbols exclude
;
$var ;output (result)
```

Note:

- Error flag if syntax error
- Same symbol to include & to exclude = to exclude

Example (UpperCase):

Example (LowerCase):

Example (Filter1):

```
Section

${StrFilter} "123abc 456DEF 7890|%#" "2" "|%"

; $R0="abcDEF|%" ;only Letters + |%

SectionEnd
```

Example (Filter2):

Example (Filter3):

Example (Filter4):

```
Section

${StrFilter} "123abcÀÁÂ 456DEFãäå 7890|%#" "+1

; $R0="123ÀÁÂ456ä7890" ;only Digits + Letters

SectionEnd
```

Example (English + Russian Letters):

Example (Word Capitalize):

```
Push "_01-PERPETUOUS_DREAMER__ - __THE_SOUND_OF_
Call Capitalize
Pop $R0
; $R0="_01-Perpetuous_Dreamer__ - __The_Sound_Of

${WordReplace} "$R0" "_" " "+*" $R0
; $R0=" 01-Perpetuous Dreamer - The Sound Of G

${WordReplace} "$R0" " " "" "{}" $R0
; $R0="01-Perpetuous Dreamer - The Sound Of Go
SectionEnd

Function Capitalize
Exch $R0
```

```
Push $0
Push $1
Push $2
${StrFilter} '$R0' '-eng' '' '' $R0
${StrFilter} '$R0' '-rus' '' ' $R0
StrCpy $0 0
loop:
IntOp $0 $0 + 1
StrCpy $1 $R0 1 $0
StrCmp $1 '' end
StrCmp $1 ' ' +5
StrCmp $1 '_' +4
StrCmp $1 '-' +3
StrCmp $1 '(' +2
StrCmp $1 '[' 0 loop
IntOp $0 $0 + 1
StrCpy $1 $R0 1 $0
StrCmp $1 '' end
${StrFilter} '$1' '+eng' '' ''
                                $1
${StrFilter} '$1' '+rus' '' ''
                                $1
StrCpy $2 $R0 $0
IntOp $0 $0 + 1
StrCpy $R0 $R0 '' $0
IntOp $0 $0 - 2
StrCpy $R0 '$2$1$R0'
goto loop
end:
Pop $2
Pop $1
Pop $0
Exch $R0
```

E.3.15 StrFilterS

• Same as StrFilter, but case sensitive.

E.3.16 VersionCompare

• Compare version numbers.

Syntax:

Example:

E.3.17 VersionConvert

• Convert version in the numerical format which can be compared.

Syntax:

```
${VersionConvert} "[Version]" "[CharList]" $var

"[Version]" ; Version
;
```

Note:

- Converted letters are separated with dot
- If character is non-digit and not in list then it will be converted to dot

Example1:

Example2:

Example3:

```
Section
```

```
${VersionConvert} "0.15c+" "abcdefghijklmnopqr
; $R0="0.15.0327"

${VersionConvert} "0.15c" "abcdefghijklmnopqrs
; $R1="0.15.03"

${VersionCompare} "$R0" "$R1" $R2
; $R2="1" version1 is newer
SectionEnd
```

Appendix F: Changelog and Release Notes

- 3.0
 - Release Notes
 - Changelog
- 3.0 Release Candidate 2
 - Changelog
- 3.0 Release Candidate 1
 - Changelog
- 3.0 Beta 3
 - Changelog
- 3.0 Beta 2
 - Changelog
- 3.0 Beta 1
 - Release Notes
 - Changelog
- 3.0 Beta 0
 - Release Notes
 - Changelog
- 3.0 Alpha 2
 - Release Notes
 - Changelog
- 3.0 Alpha 1
 - Release Notes
 - Changelog
- 3.0 Alpha 0
 - Release Notes
 - Changelog
- 2.51
 - Changelog
- 2.50
 - Changelog
- 2.49
 - Changelog
- 2.48
 - Changelog

- 2.47
 - Release Notes
 - Changelog
- 2.46
 - Changelog
- 2.45
 - Release Notes
 - Changelog
- 2.44
 - Changelog
- 2.43
 - Release Notes
 - Changelog
- 2.42
 - Release Notes
 - Changelog
- 2.41
 - Changelog
- 2.40
 - Changelog
- 2.39
 - Changelog
- 2.38
 - Changelog
- 2.37
 - Changelog
- 2.36
 - Release Notes
 - Changelog
- 2.35
 - Changelog
- 2.34
 - Release Notes
 - Changelog
- 2.33
 - Release Notes
 - Changelog

- 2.32
 - Release Notes
 - Changelog
- 2.31
 - Release Notes
 - Changelog
- 2.30
 - Release Notes
 - Changelog
- 2.29
 - Release Notes
 - Changelog
- 2.28
 - Changelog
- 2.27
 - Release Notes
 - Changelog
- 2.26
 - Release Notes
 - Changelog
- 2.25
 - Changelog
- 2.24
 - Changelog
- 2.23
 - Changelog
- 2.22
 - Release Notes
 - Changelog
- 2.21
 - Changelog
- 2.20
 - Changelog
- 2.19
 - Changelog
- 2.18
 - Changelog

- 2.17
 - Changelog
- 2.16
 - Release Notes
 - Changelog
- 2.15
 - Changelog
- 2.14
 - Release Notes
 - Changelog
- 2.13
 - Release Notes
 - Changelog
- 2.12
 - Changelog
- 2.11
 - Release Notes
 - Changelog
- 2.10
 - Changelog
- 2.09
 - Changelog
- 2.08
 - Release Notes
 - Changelog
- 2.07
 - Release Notes
 - Changelog
- 2.06
 - Changelog
- 2.05
 - Release Notes
 - Changelog
- 2.04
 - Changelog
- 2.03
 - Release Notes

- Changelog
- 2.02
 - Changelog
- 2.01
 - Release Notes
 - Changelog
- 2.0
 - Release Notes
 - Changelog
- 2.0 Release Candidate 4
 - Changelog
- 2.0 Release Candidate 3
 - Changelog
- 2.0 Release Candidate 2
 - Changelog
- 2.0 Release Candidate 1
 - Release Notes
 - Changelog
- 2.0 Beta 4
 - Release Notes
 - Changelog
- 2.0 Beta 3
- 2.0 Beta 2
- 2.0 Beta 1
- 2.0 Beta 0
- 2.0 Alpha 7
- 2.0 Alpha 6
- 2.0 Alpha 5
- 2.0 Alpha 4
- 2.0 Alpha 3
- 2.0 Alpha 2
- 2.0 Alpha 1
- 2.0 Alpha 0
- Older Versions

F.1 3.0

Released on July 24th, 2016

F.1.1 Release Notes

Long-awaited version 3 of NSIS is finally ready. It comes with optional Unicode support, still works on all the same versions of Windows, supports Windows 10, and adds a lot of other small features and fixes. Huge thanks goes out to Anders and Wizou for making this happen and Jim Park for the original patch.

F.1.2 Changelog

F.1.2.1 Minor Changes

• !finalize command now supports the same compare operators as !system (bug #1148)

F.1.2.2 Translations

• PortugueseBR fixes (bug #1149)

F.2 3.0 Release Candidate 2

Released on July 8th, 2016

F.2.1 Changelog

F.2.1.1 Major Changes

- Fixed LogicLib nested \${Select} bug
- !makensis command must force Unicode output to be UTF-8 (bug #1147)

F.2.1.2 Minor Changes

- Zip2Exe aborts if the zip file is encrypted (bug #1141)
- Added LogicLib & operator
- NSIS Menu POSIX fixes (bug #1144)

F.2.1.3 Translations

- PortugueseBR fixes (deguix)
- Minor tweaks to Danish.nlf (scootergrisen) (bug #1140)

F.3 3.0 Release Candidate 1

Released on April 3nd, 2016

F.3.1 Changelog

F.3.1.1 Major Changes

- Fixed !appendfile on POSIX and LogicLib \${Switch}
- More security hardening to prevent dll hijacking

F.3.1.2 Minor Changes

- Documentation improvements
- The name of the uninstaller created in %TEMP%\~nsu.tmp has changed from %UNIQUE%u .exe to Un %UNIQUE%.exe
- Various warning and error improvments

F.3.1.3 Translations

Added missing MULTIUSER_* entries to Japanese.nsh (patch #266)

F.4 3.0 Beta 3

Released on December 26th, 2015

F.4.1 Changelog

F.4.1.1 Major Changes

- RequestExecutionLevel now defaults to admin
- LoadLibrary security hardening to prevent dll hijacking (bug #1125)
- FileReadUTF16LE now skips the optional BOM at the start of a file

F.4.1.2 Minor Changes

- Fixed System plugin GUID type output bug on Win98
- FileWriteUTF16LE can add a BOM with the /BOM switch
- CreateShortcut icon index can now be larger than 255 (bug #1123)
- !system and !execute now provide an empty StdIn pipe to work around bugs in some Windows utilities
- Added support for 0o octal radix prefix on number literals in the preprocessor
- The single parameter version of !if now also supports floats
- Preprocessor now warns when invalid floating point numbers are used in math operations
- MakeNSISW now uses WinInet when checking for updates

F.4.1.3 Translations

- Brazilian Portuguese updated (Felipe) (patch #263)
- Polish updated (Paweł Porwisz)

F.5 3.0 Beta 2

Released on August 4th, 2015

F.5.1 Changelog

F.5.1.1 Major Changes

• Preliminary Windows 10 support

F.5.1.2 Minor Changes

- Added !appendfile /RawNL switch
- Added PESubsysVer attribute
- Exec[Wait] sets the CREATE_DEFAULT_ERROR_MODE flag when creating a process
- Fixed 4+ TiB freespace calculation bug (bug #1115)
- Fixed CreateShortcut /NoWorkingDir parsing bug (bug #1110)
- Fixed minor issues in the Pascal NSIS plug-in SDK and removed the extrap global variable
- nsDialogs and InstallOptions now use the system link color

F.5.1.3 Build System

- Linux stdcall warning fix (patch #261)
- Linux test-scripts fix (patch #260)
- Win64 fixes (including patch #258, bug #1105)
- Visual C 2012 fixes

F.6 3.0 Beta 1

Released on October 6th, 2014

F.6.1 Release Notes

 This release addresses a lot of build issues, reintroduces the nightly builds and adds automatic nightly test execution

F.6.2 Changelog

F.6.2.1 Major Changes

- !insertmacro allows macro recursion (RFE #497)
- Added !makensis command
- Added new MUI2 bitmap stretch modes, *_NOSTRETCH is now deprecated (RFE #521)
- InitiateShutdown() is used to reboot the machine if available (patch #247)
- Added PPO and SafePPO preprocess-only compiler switches
- MakeNSIS WM_COPYDATA messages now use the QH_OUTPUTCHARSET encoding with CP_ACP as the default for compatibility with old IDEs.

F.6.2.2 Minor Changes

- Added IsWow64 to x64.nsh
- Added PEDIICharacteristics attribute
- Added System::Call direct register memory access type. (patch #249)
- Added WX compiler switch
- Allow skipping ExDLL build with SKIPPLUGINS ((patch #254))
- Changed default DllCharacteristics to TS AWARE+NO SEH+NX COMPAT+DYNAMIC BASE

- Fixed Mac OS X builds (patch #253 and bug #1085)
- Fixed POSIX !searchparse bug (patch #251)
- Fixed !macroundef
- Fixed test-code build target on POSIX and enabled it in nightly builds (bug #1098)
- Fixed Visual Studio 2012 builds

F.6.2.3 Translations

• Added Armenian (Hrant Ohanyan)

F.7 3.0 Beta 0

Released on May 11th, 2014

F.7.1 Release Notes

 POSIX builds work again! Please let us know if your favorite platform is still having build issues. We mainly test on Linux.

F.7.2 Changelog

F.7.2.1 Major Changes

- Basic AMD64 System::Call support
- Fixed POSIX builds

F.7.2.2 Minor Changes

- Added CreateShortcut /NoWorkingDir parameter
- Added Int<32|64|Ptr><Op|Cmp[U]> helper macros to Util.nsh
- Added P<, P<=, P=, P<>, P>= and P> LogicLib ptrdiff t tests
- Try harder to find duplicate strings in string block (bug #1088)
- !system will decode child output as OEMCP if GetConsoleOutputCP() == GetOEMCP()
- !system and !execute can store the exit code in a define
- !execute supports comparing the exit code with the same syntax as
 !system
- Preprocessor supports on and ob radix prefix on number literals

F.8 3.0 Alpha 2

Released on December 24th, 2013

F.8.1 Release Notes

It's the holiday gift you've been waiting for!

F.8.1.1 Known Issues

- The POSIX build is getting much closer, but still currently broken
- LangDLL doesn't display localized language name with Unicode false

F.8.2 Changelog

F.8.2.1 Major Changes

- !include defaults to UTF-8 after unicode True
- Preprocessor does not parse all branches and will not validate code that is never executed (bug #1086)

F.8.2.2 Minor Changes

- FileRead in Unicode installers can handle DBCS, conversion output is limited to UCS-2.
- FileRead in Unicode installers now uses the Unicode replacement character (U+FFFD) for invalid characters and not '?'.
- FileReadByte no longer performs a Unicode conversion on non-ASCII characters
- Windows 8.1 & 2012R2 support in WinVer.nsh
- Fixed !define Unicode null terminator bug (bug #1079)
- Unicode stubs create WCHAR richedit controls (bug #1080)
- Fixed MakeNSISW default pushbutton and tab order

• Fixed incorrect page count in compiler statistics

F.9 3.0 Alpha 1

Released on July 14th, 2013

F.9.1 Release Notes

F.9.1.1 Known Issues

- The POSIX build is currently broken
- LangDLL doesn't display localized language name with Unicode false
- FileRead may return '?' for MBCS letters with Unicode true

F.9.2 Changelog

F.9.2.1 Major Changes

- Default verbosity is /V3 without logo, MakeNSISW still uses /V4.
- ManifestSupportedOS added support for the Windows 8.1 GUID and it is set by default to avoid GetVersionEx compatibility behavior.

F.9.2.2 Minor Changes

- Fixed !finalize %1
- Fixed !searchparse (bug #1073)
- Made installers always respect /S on command line, even when installer file can't be read (bug #1076)
- Minor MakeNSISW fixes and tweaks
- Added Zip2Exe Unicode checkbox

F.10 3.0 Alpha 0

Released on May 19th, 2013

F.10.1 Release Notes

 MakeNSIS can now generate ANSI and Unicode installers. Source files can be UTF8SIG, UTF16BOM or traditional MBCS text files (which are converted to Unicode with ACP unless you specify a different codepage). The default plugins are now stored in subfolders based on their CPU target and character set encoding.

F.10.1.1 Known Issues

- The POSIX build is currently broken
- LangDLL doesn't display localized language name with Unicode false
- FileRead may return '?' for MBCS letters with Unicode true

F.10.2 Changelog

F.10.2.1 Major Changes

- Added the Unicode attribute (RFE #1238132, patch #1795257)
- Added support for Windows 8 and Windows Server 2012
- MakeNSIS can read UTF8SIG and UTF16BOM script files (RFE #2026892)
- All NLF and NSH language files are stored as UTF-16LE (RFE #1879642)
- Scripts can control the SupportedOS list in the application manifest using ManifestSupportedOS (bug #2725883, RFE #3020103)
- Installers can claim DPI-awareness with ManifestDPIAware (bug #2897169)
- The index of the changed section is stored in \$0 during

- .onSelChange callbacks (RFE #1634936)
- \${U+1}...\${U+10FFFF} are treated as a Unicode character unless there is already a define with that name (RFE #2084797)

F.10.2.2 Minor Changes

- Added !getdllversion (bug #2809308)
- Added !appendfile /CHARSET parameter
- Added !if support for /fileexists, case sensitive comparisons, hex and float
- Added more definitions to WinMessages.nsh
- %temp%\Low will be used if the installer cannot write to %temp% nor %windir%\Temp (bug #2909242, patch #2912824)
- Added \${NSIS_PACKEDVERSION}, the NSIS version packed as a hex number (patch #2680832)
- Added \${__MACRO__} and \${__COUNTER__} predefines
- Added UnsafeStrCpy instruction, it can write to special variables like \$PLUGINSDIR
- Added VIFileVersion
- Added !finalize for post-build commands (like signing the installer)
- Plugins in \${NSISDIR}\Plugins have to be reserved with ReserveFile /plugin
- MakeNSIS /NOTIFYHWND uses a (optional) new event name to abort compilation, see build.cpp/h for details.
- Reduced !include/!insertmacro recursion stack usage (bug #3067954)
- Fixed minor MakeNSIS leaks (bug #3474662)
- Fixed MUI_FINISHPAGE_TITLE_3LINES padding when reboot is required (bug #3400067, bug #3408407)
- Fixed removal of read only attribute even if Delete or RMDir fails (bug #3072159)
- Fixed \${__SECTION__} in hidden sections (RFE #1787648)
- Various documentation fixes (bug #3063566, bug #3567313, patch #3307144)
- Lots of other small fixes...

F.10.2.3 Translations

- Changed LANGFILE macro in LangFile.nsh
- Warnings are now displayed for missing strings (LANGFILE_INCLUDE_WITHDEFAULT)
- Added Georgian (David Huriev)
- Added Pashto (Pakhtosoft)
- Added Scottish Gaelic (GunChleoc)
- Fixed Korean MUI_[UN]TEXT_FINISH_INFO_* (bug #3541515)

F.11 2.51

Released on April 1st, 2016

F.11.1 Changelog

F.11.1.1 Minor Changes

- More security hardening to prevent dll hijacking
- Backported support for system 'p' type and logiclib P and Z tests

F.11.1.2 Translations

Added missing MULTIUSER_* entries to Japanese.nsh (patch #266)

F.12 2.50

Released on December 26th, 2015

F.12.1 Changelog

F.12.1.1 Major Changes

• RequestExecutionLevel now defaults to admin

F.12.1.2 Minor Changes

• Preload certain system libraries to prevent dll hijacking (bug #1125)

F.13 2.49

Released on December 16th, 2015

F.13.1 Changelog

• Fixed CreateShortcut failure on Vista caused by security changes in 2.47

F.14 2.48

Released on December 10th, 2015

F.14.1 Changelog

• Fixed CHM (bug #1129)

F.15 2.47

Released on December 8th, 2015

F.15.1 Release Notes

• This is a maintenance and security release

F.15.2 Changelog

F.15.2.1 Major Changes

- LoadLibrary security hardening to prevent dll hijacking (bug #1125)
- InitiateShutdown() is used to reboot the machine if available (patch #247)

F.15.2.2 Minor Changes

- Fixed 4+ TiB freespace calculation bug (bug #1115)
- Windows 8 & 2012 support in WinVer.nsh

F.16 2.46

Released on December 6th, 2009

F.16.1 Changelog

F.16.1.1 Minor Changes

- Fixed !searchparse (bug #2803622)
- Fixed Vista error message when canceling uninstaller (bug #2803097)
- Resolve warning for LogicLib's endless Do..Loop (bug #2849872)
- StartMenu: fixed random number prefixes when clicking on an empty spot (bug #2810188)
- Updates to NSIS.pas for plug-in development (patch #2802794)

F.16.1.2 Translations

- PortugueseBR typo (patch #2826598)
- Removed double spaces in Italian (bug #2873598)

F.16.1.3 Build System

Fixed mingw builds of Math plug-in (patch #2835731)

F.17 2.45

Released on June 6th, 2009

F.17.1 Release Notes

• Support for Microsoft's upcoming release of Windows 7 has been added based on RC1 testing and documentation. Please report any related issues and don't forget to use RequestExecutionLevel.

F.17.2 Changelog

F.17.2.1 Major Changes

- Added support for Windows 7 installers use RequestExecutionLevel, just like with Vista (bug #2725883)
- Added WinVer.nsh Windows 7 and Windows 2008 R2 support
- Installers now identify as Terminal Services aware (IMAGE_DLLCHARACTERISTICS_TERMINAL_SERVER_AWARE) so \$WINDIR will no longer be under the user's profile when installing on Terminal Services
- Less UAC annoyance in Add/Remove control panel (bug #2697027)

F.17.2.2 Minor Changes

- Added wildcard and /nonfatal support for !delfile (RFE #1505425)
- Fixed !searchparse (bug #2680110)
- Fixed input validation for CreateFont (bug #2801024)
- Fixed NSIS Menu handling of working directory (bug #2781948)
- Fixed Vista issues with shell folders (\$DOCUMENTS, \$PROFILE, etc.) on root directories (bug #2138075)
- Minor documentation improvements (including bug #2705878, bug #2606525)
- Use SW SHOWDEFAULT for ExecShell by default (bug #2796189)

F.17.2.3 Modern UI

- Call finish page show function only after the page is fully initialized (bug #2720968)
- Fixed translation of uninstaller directory page (bug #2690112)
- Fixed installation type text position in components page (patch #2801317)
- Fixed missing MUI_INNERTEXT_COMPONENTS_DESCRIPTION_TITLE (bug #2788620)

F.17.2.4 Translations

- Updated Indonesian (patch #2790571)
- Updated Portuguese BR (patch #2642542)

F.17.2.5 Build System

Fixed some GCC warnings

F.18 2.44

Released on February 21st, 2009

F.18.1 Changelog

F.18.1.1 Major Changes

 Fixed a bug introduced in 2.43 causing script build issues with some icons (bug #2572035)

F.18.1.2 Minor Changes

- Fixed nsDialogs CreateTimer documentation (bug #2595565)
- Global labels didn't work when declared in unused functions (bug #2593369)

F.19 2.43

Released on February 5th, 2009

F.19.1 Release Notes

• A few minor changes were made to the new plug-in API that break backward compatibility. Header and library paths were changed and RegisterPluginCallback has changed its return value.

F.19.2 Changelog

F.19.2.1 Minor Changes

- Added WinCore.nsh, WinDef.nsh, WinError.nsh, WinNT.nsh and WinUser.nsh for more useful Windows definitions
- Fixed a crash caused by !packhdr compressing resources (bug #2533431)
- Minor documentation improvements (including bug #2564005)
- Modern UI 2: Fixed MUI_DIRECTORYPAGE_BGCOLOR (bug #2494528)

F.19.2.2 Utilities and Plug-ins

- Banner: Fixed installer showing on the background when Banner was used in .onInit
- MakeNSISW: Added Ctrl+K hotkey for build cancelation (RFE #2557392)
- nsDialogs: Added NSD SetIcon (patch #2500960)

F.19.2.3 Translations

- Added Esperanto
- Bulgarian fixes

F.19.2.4 Plug-in API

- Added VS2008 project files
- Better installation of header and library files under POSIX see INSTALL file for more information
- Header and library files were moved to a subdirectory named *nsis* to prevent collisions
- RegisterPluginCallback now returns int instead of BOOL for a wider range of error reporting

F.19.2.5 Build System

- Added fink's mingw prefixes (bug #2495138)
- Fixed BOOL build issue on OS X (bug #2497290)
- Fixed iconv dependency detection on OS X (bug #2494539)
- Fixed Solaris builds (patch #2497172)

F.20 2.42

Released on December 20th, 2008

F.20.1 Release Notes

- Merry Christmas and a happy Hanukkah!
- Plug-in developers should check out the new plug-in API in Examples\Plugin and convert their plug-ins, especially in case they require staying loaded.

F.20.2 Changelog

F.20.2.1 Major Changes

- Deprecated /NOUNLOAD and SetPluginsUnload to make scripts simpler and safer (patch #1912699)
- Useful header functions no longer require usage declaration and different syntax for uninstaller functions
- Revamped plug-in API now comes in the form of pluginapi.lib, API version information and more common functions (patch #2359978)

F.20.2.2 Minor Changes

- Added !searchreplace preprocessor command for compiletime text search/replaces
- Added support for registration of EXE COM servers (RFE #2315740)
- Minor documentation improvements (including bug #2386821)
- nsDialogs: Added timer support (patch #2135855)
- nsDialogs: Added progress bar support
- WinVer.nsh: Added IsServer, IsWin2003R2, IsStarterEdition, OSHasMediaCenter and OSHasTabletSupport
- WinVer.nsh: Fixed 95/NT4 ambiguity (bug #2053642)
- WinVer.nsh: Proper Windows XP x64 detection (bug #2053700)

• WinVer.nsh: Windows 2008 detection support (RFE #1949260)

F.20.2.3 Translations

• Minor English grammar improvements (bug #2323452)

F.20.2.4 Build System

• Full System compatibility with GCC (patch #2193442)

F.21 2.41

Released on November 20th, 2008

F.21.1 Changelog

F.21.1.1 Minor Changes

- Fixed LangDLL memory leaks for invalid input (bug #1939573)
- Fixed uninstaller generation on big-endian systems (bug #2166401, bug #2167958)

F.21.1.2 Translations

- Brazilian Portuguese corrections
- Consistency fixes for Simple Chinese (patch #2189117)

F.21.1.3 Build System

- Added support for SCons 1.1.0
- Fixed directory and components page text issues on VC8 builds (patch #1982084)
- System's Resource.dll now built from source

F.22 2.40

Released on October 10th, 2008

F.22.1 Changelog

F.22.1.1 Major Changes

 Service pack macros added to WinVer in version 2.39 now work (bug #2070708, patch #2095363)

F.22.1.2 Minor Changes

- Added initial folder selection option for nsDialogs::SelectFileDialog (patch #2016003)
- Fixed GetParameters handling of MBCS characters (bug #2067946)
- Fixed incorrect \${NSD OnBack} documentation (bug #2059651)
- Modern UI 2: Set focus to checkboxes on the finish page (bug #2110357)

F.22.1.3 Translations

- Portuguese corrections (bug #2086988)
- Simplified Chinese corrections (patch #2056906)

F.23 2.39

Released on August 16th, 2008

F.23.1 Changelog

F.23.1.1 Major Changes

- Added !define /file and !searchparse (patch #2016254)
- Added service pack macros (AtLeastServicePack, IsServicePack, AtMostServicePack) to WinVer.nsh (patch #2036802)

F.23.1.2 Minor Changes

- Added more nsDialogs list box handling macros (patch #2041919)
- Added Unicode version compatible System string type (patch #2025721, RFE #1961307)
- Fixed nsDialogs atom leak (bug #2053522)
- Minor documentation improvements
- Show hand cursor for nsDialogs links (patch #2004129)

F.24 2.38

Released on July 12th, 2008

F.24.1 Changelog

F.24.1.1 Major Changes

 Fixed a bug in nsDialogs that caused it to pollute the stack if callbacks are not set for each control (bug #2013317)

F.24.1.2 Minor Changes

- Added IfNotThen support for LogicLib (patch #1990761)
- Added support for NTFS mounts points on the directory page (bug #1946112)
- Fixed branding image control detection on Debian, due to improper identification of static controls (bug #1951417)
- Log actual creation of directories and not just failures and final directory (patch #1992325)
- Made log close when LogSet off is used (patch #1986692)
- Minor documentation improvements (including bug #1990955)
- MultiUser: Fixed MULTIUSER_INSTALLMODE_INSTDIR handling for the uninstaller

F.24.1.3 Modern UI

- Document page leave callback function (bug #1964719)
- Fixed MUI_FINISHPAGE_CANCEL_ENABLED
- Fixed wasted variable warnings (bug #1995024)

F.24.1.4 Translations

- Fixed French translation of MUI_UNTEXT_FINISH_INFO_REBOOT (patch #1967032)
- Removed non-ANSI characters from French display name (bug #1979491)
- Updated Slovenian (patch #2014106)

F.25 2.37

Released on May 3rd, 2008

F.25.1 Changelog

F.25.1.1 Major Changes

- Fixed a bug introduced in 2.32 that caused blurry icons on Windows versions prior to XP (bug #1956350)
- Use \$PROGRAMFILES as a default for \$PROGRAMFILES64 instead of "C:\Program Files" (bug #1947702)

F.25.1.2 Minor Changes

- Automatically select language in case there is only one available choice (bug #1939571)
- MultiUser: Fixed build errors with manual inclusion of StrFunc.nsh
- Support for compression of files up to 2GB (patch #1948700)

F.25.1.3 Utilities and Plug-ins

- Fixed \${NSD OnBack} (bug #1947388)
- Fixed nsDialogs::SelectFileDialog return value for user cancelation (bug #1955803)
- Fixed possible BgImage crash when calling BgImage::Destroy more than once (patch #1951248)
- Minor documentation improvements

F.25.1.4 Translation

- Brazilian Portuguese updates
- Polish updates (patch #1927421)

- Slovak updates (patch #1939669)
- Spanish updates

F.25.1.5 Build System

• SCons requirement upgraded to 0.98

F.26 2.36

Released on March 29th, 2008

F.26.1 Release Notes

 nsDialogs is picking up its pace and offers lots of new macros in this release. Keep the patches and requests coming!

F.26.2 Changelog

F.26.2.1 Major Changes

 nsExec: Support for x64 disabled redirection (RFE #1778973, bug #1889317)

F.26.2.2 Modern UI

- Added missing MUI_UNFUNCTION_DESCRIPTION_BEGIN and MUI_UNFUNCTION_DESCRIPTION_END
- Fixed start menu page setting the error flag (bug #1891106)

F.26.2.3 nsDialogs

- NSD_AddStyle and NSD_AddExStyle for easy customization (patch #1900588)
- NSD_CB_AddString, NSD_CB_SelectString, NSD_LB_AddString and NSD_LB_SelectString for easier handling of combo and list boxes
- NSD_CreateNumber and NSD_SetTextLimit for limited input fields
- NSD_SetImage, NSD_SetStretchedImage, NSD_ClearImage and NSD_FreeImage for image handling
- NSD_SetState, NSD_GetState, NSD_Check and NSD_Uncheck for handling of check boxes and radio buttons (patch #1900588)

- NSD_SetText for easily setting control's text
- OnClick support for labels (bug #1908732)
- Support for edit box change notification in combo box

F.26.2.4 Minor Changes

- Fixed exception handling of UPX compressed icons (bug #1896500)
- Fixed InstallDirRegKey example in bigtest.nsi
- Fixed Memento's documentation to include \${MementoSectionDone}
- Fixed rare decompression error with 0x4001 bytes of compressed data (bug #1874297)
- LangFile.nsh: Added LANGFILE_INCLUDE_WITHDEFAULT to load language file with defaults being obtained from a default file. This replaces the LANGFILE_DEFAULT setting and makes it easier to use LangFile.nsh for multiple sets of languages file without having to care about the different LANGFILE_DEFAULT settings.
- Minor documentation improvements (including bug #1891266, bug #1894033, bug #1896803)
- MultiUser: Fixed error when a custom installation mode initialization function is used only for the installer
- MultiUser: Fixed default installation mode based on registry key (bug #1913029)
- MultiUser: Fixed no uninstaller handling (MULTIUSER NOUNINSTALL)

F.26.2.5 Utilities and Plug-ins

- MakeNSISW: Avoid opening executables with identical folder names when opening script's folder (bug #1891066)
- nsExec: Fixed return code handling so 259 won't cause an infinite loop (bug #1909458)
- VPatch: Added MD5 functions (RFE #1900226)

F.26.2.6 Translation

- Albanian updates (patch #1919360)
- French fixes (patch #1916564)
- Polish updates (patch #1894983)

F.26.2.7 Build System

- Fixed build of NSIS Menu with wxGTK (bug #1900233)
- Use CRLF for DSW and DSP files to avoid corruption messages
- Various build fixes for GCC 4.3, amd64 and Linux in general

F.27 2.35

Released on February 8th, 2008

F.27.1 Changelog

F.27.1.1 Major Changes

- Added MultiUser installer configuration for multi-user Windows environments
- More user friendly corrupted installer message (RFE #1795426)
- Switched all examples to Modern UI 2

F.27.1.2 Minor Changes

- Fixed uninitialized memory leaking into resources (bug #1874297)
- Minor documentation improvements (including bug #1861941, bug #1883917)
- Modern UI 2: Fit images in welcome/finish page by default and fix support for MUI_(UN)WELCOMEFINISHPAGE_BITMAP_NOSTRETCH (bug #1875945)
- Modern UI 2: Fixed default finish page reboot selection (bug #1864690)
- Modern UI 2: Fixed MUI_STARTMENU_GETFOLDER (bug #1864507)
- Modern UI 2: Fixed unreferenced mui.StartMenuPage.Create warning (bug #1861944)
- New simplified code for installation of VisualBasic 6 runtime
- Simplified NSIS Menu providing easier access to all documentation

F.27.1.3 Utilities and Plug-ins

Added keyboard cues support for LINK controls in nsDialogs (patch

#1871856)

• Fixed a rare nsDialogs crash on Windows 98 with non-standard builds of nsDialogs (bug #1889720)

F.28 2.34

Released on December 24th, 2007

F.28.1 Release Notes

 Thanks to user input on Modern UI 2, it is now ready for mass consumption at no better time than the holiday season. The installer for NSIS itself now exploits the wonders of MUI2 and nsDialogs and so should you.

F.28.2 Changelog

F.28.2.1 Major Changes

 Modern UI 2: Simpler code, easier to extend and makes use of the faster nsDialogs

F.28.2.2 Minor Changes

- Added IfNot support for LogicLib (patch #1846785)
- Added some deprecation messages in MUI2 (bug #1784470)
- Allow overwrite of LANGFILE_*_NAME defines in new LangFile.nsh (bug #1848952)
- Better RTL support in instfiles and components page (bug #1841573)
- Don't warn of comments containing line-continuation character when it has no effect (bug #1701051, RFE #1686589)
- Fixed deletion of files with relative paths, a.k.a the EVE bug (bug #1851273)
- Fixed input verification of GetCurInstType and GetFullPathName
- Fixed WordFind handling of MBCS characters (bug #1852141)
- Minor documentation improvements (including bug #1842326)
- Use CRLF in examples (bug #1835866)

F.28.2.3 Utilities and Plug-ins

- Added missing stack handling in nsDialogs examples
- Added NSD_GetState for checkboxes and radio buttons, NSD_SetFocus, NSD_CreatePassword and NSD_CreateDropList (patch #1848940)
- Fixed documentation of nsDialogs::SelectFolderDialog (bug #1841120)
- Fixed NSD_CreateComboBox so it'd work like InstallOptions (bug #1851136)

F.28.2.4 Translations

- Bulgarian updates
- German fixes

F.28.2.5 Build System

- Fixed build issues on Mac OS (bug #1851365)
- Fixed endianity issues introduced in version 2.32 (bug #1851365)

F.29 2.33

Released on November 17th, 2007

F.29.1 Release Notes

This is a quick fix release for a bug, introduced in 2.32, that
prevented the welcome page to display properly on Windows 9x.
 The problem was caused by the move from CVS to Subversion that
messed with the line-breaks format of the INI files.

F.29.2 Changelog

F.29.2.1 Major Changes

 Modern UI: Fixed a bug, introduced in 2.32, that caused display problems for the welcome and finish pages on Windows 9x (bug #1831677)

F.29.2.2 Minor Changes

 Modern UI: Allow MUI_LANGDLL_ALLLANGUAGES to be defined after insertion of MUI_LANGUAGE macro

F.29.2.3 Utilities and Plug-ins

- Made nsExec's internal process always exit cleanly to avoid false return values and delays (reported in the forum)
- nsDialogs: Added OnClick support for static controls
- VPatch: Fixed VPatchFile macro path handling (bug #1829540)

F.29.2.4 Translations

• Spanish updates

F.29.2.5 Build System

• Separate strip options for win32 and cross-platform binaries

F.30 2.32

Released on November 9th, 2007

F.30.1 Release Notes

 Vista PNG icons are now supported without the need for any special tricks.

F.30.2 Changelog

F.30.2.1 Major Changes

- Added support for mismatching installer and uninstaller icons
- NSIS source code is now hosted on Subversion instead of CVS

F.30.2.2 Minor Changes

- Fixed synchronization that could allow the user to cause extraction errors in rare cases (reported in the forum)
- Fixed syntax validation of Var
- Minor documentation improvements (including bug #1811876, bug #1819946)
- Modern UI 2 beta: Fixed 3-line title on welcome/finish page
- Modern UI 2 beta: Fixed components page description and lock
- Modern UI 2 beta: Fixed default language selection from registry
- Modern UI 2 beta: Fixed MUI_DIRECTORYPAGE_BGCOLOR

F.30.2.3 Utilities and Plug-ins

nsDialogs: Added link support

F.30.2.4 Translations

- Bulgarian corrections
- Fixed Japanese and Norwegian Nynorsk MUI line breaks (bug #1817289)

F.30.2.5 Build System

• Check compiler's -m32 flag before the linker's to avoid gcc segfault

F.31 2.31

Released on September 29th, 2007

F.31.1 Release Notes

 Please continue to report any issues and suggestions regarding Modern UI 2. Input for the previous version was very helpful in improving MUI2.

F.31.2 Changelog

F.31.2.1 Minor Changes

- Added BST * definitions to WinMessages.nsh (bug #1792422)
- Added SetRegView lastused
- Minor documentation improvements
- Library will now revert registry view settings (SetRegView) after it's finished
- Modern UI: Fixed empty welcome page with Chinese (bug #1786899)
- Modern UI: Fixed finish button text (bug #1789492)
- Modern UI: Fixed uninstaller comportments page text (bug #1793811)
- Modern UI: Fixed unused function warnings
- Modern UI 2 beta: Fixed header bitmap issues, including NOSTRETCH and compilation errors
- Modern UI 2 beta: Fixed language selection dialog
- Modern UI 2 beta: Fixed reboot finish page
- Modern UI 2 beta: Fixed unloading of nsDialogs on finish page
- Modern UI 2 beta: Fixed unused variable warnings

F.31.2.2 Utilities and Plug-ins

• MakeNSISW: Fixed command line parsing error (bug #1796053)

F.31.2.3 Translations

- Korean corrections
- Persian improvements (patch #1776386)
- Romanian improvements (patch #1783853)

F.31.2.4 Build System

• Fixed build failures with MinGW's GCC 4 (bug #1800834)

F.32 2.30

Released on August 25th, 2007

F.32.1 Release Notes

- This release includes a beta of Modern UI 2. Please report any issues and suggestions.
- The format of Modern UI language files has been modified to better support external strings. Private language files should be converted (and submitted as a patch in the spirit of open source).

F.32.2 Changelog

F.32.2.1 Major Changes

- Fixed a bug introduced in version 2.29 that caused invalid \$SMPROGRAMS value on Windows 98 (bug #1766268)
- Modern UI 2 beta: Simpler code, easier to extend and makes use of nsDialogs

F.32.2.2 Minor Changes

- Added InstallOptions.nsh with relevant usage macros, based on the old Modern UI macros
- Added LangFile.nsh allowing creation of langauge files that can be included with a single command with defaults for missing strings (when LANGFILE DEFAULT is set)
- Added LB ERR and CB ERR to WinMessages.nsh (bug #1771644)
- Fixed CreateDirectory logging (patch #1768584)
- Fixed escaping of quotes with \$\ in macros (bug #1713708)
- Minor documentation improvements
- Modern UI 1.80: New language files structure based on LangFile.nsh
- Modern UI 1.80: MUI LANGDLL DISPLAY now reads previous

settings on silent installations as well

 Modern UI 1.80: Added MUI_CUSTOMFUNCTION_MOUSEOVERSECTION (patch #1762003)

F.32.2.3 Utilities and Plug-ins

 Various nsDialogs improvements including RTL support, more macros in nsDialogs.nsh and initial documentation

F.32.2.4 Translations

- Catalan returns
- Danish corrections
- Slovak corrections (patch #1762627)

F.32.2.5 Build System

- More HPUX fixes (bug #1755148, bug #1753063, bug #1758873, patch #1758863)
- Ignore known failing tests by default on POSIX

F.33 2.29

Released on July 14th, 2007

F.33.1 Release Notes

 The most notable addition in this release is nsDialogs - a faster and far more capable replacement for InstallOptions. It allows creating controls of any type directly from the script and removes the need to mess with slow INI files. It integrates directly into the script by calling functions for notification, including change notification for edit boxes.
 Speak up if you have any suggestions/comments/patches for it.

F.33.2 Changelog

F.33.2.1 Major Changes

• Added nsDialogs - a replacement for InstallOptions

F.33.2.2 Minor Changes

- Disable Windows error messages for the entire installer and not just for an incomplete list of fail-points (bug #1741061)
- Fixed incorrect large version handling of GetFileVersion (bug #1742255, patch #1742562)
- Fixed the ReverseSection macro in Sections.nsh (bug #1742793)
- Minor documentation improvements
- Simplified REG_MULTI_SZ reader
- Use SHGetFolderPath where available to better support all users' folders

F.33.2.3 Utilities and Plug-ins

• Banner: Avoid possible hangs when called from page callbacks (bug

#1743801)

- NSISdl: added /TRANSLATE2 for more "translator-friendly" strings (patch #1656076)
- NSISdl: support downloads over 2gb (patch #1723131)

F.33.2.4 Translations

• Added International Spanish

F.33.2.5 Build System

• Added support for HPUX (patch #1714416)

F.34 2.28

Released on June 8th, 2007

F.34.1 Changelog

F.34.1.1 Major Changes

 Added a workaround for a Windows 2000 bug that caused infinite loops when hitting a key on the instfiles page (bug #1733692)

F.34.1.2 Minor Changes

- Improved compiler error strings (patch #1722147)
- Modern UI: added MUI_STARTMENUPAGE_BGCOLOR and MUI_DIRECTORYPAGE_BGCOLOR (patch #1706187)
- Modern UI: added MUI_LANGDLL_ALLLANGUAGES that forces the language selection dialog to display all languages (patch #1724876)

F.34.1.3 Utilities and Plug-ins

- InstallOptions: corrected documentation of TxtColor (bug #1716614)
- NSISdl: added compatibility with some buggy servers that don't close the connection (bug #1713562)
- NSISdl: fixed 100% CPU usage during domain resolution (bug #1713560)

F.34.1.4 Translations

Valencian and Catalan are no longer installed (patch #1558822)

F.34.1.5 Build System

- Added IGNORETESTS for ignoring certain test files
- Added wxWidgets to the requirement list
- System now builds with GCC, but without System::Get and System::Call (patch #1711089)

F.35 2.27

Released on May 5th, 2007

F.35.1 Release Notes

• This is a quick-fix release that addresses a bug introduced in the last version, 2.26, that caused \$PROGRAMFILES and \$COMMONFILES to translate into random strings in the uninstaller.

F.35.2 Changelog

F.35.2.1 Major Changes

 Fixed a bug introduced in 2.26 that made \$PROGRAMFILES and \$COMMONFILES unavailable in the uninstaller

F.35.2.2 Minor Changes

- Disable the X button on InstallOptions pages when CancelEnabled is
- Fixed incomplete MessageBox usage line (bug #1709460)

F.35.2.3 Build System

• Added TOOLSET option to allow selection of specific build tools (e.g. scons TOOLSET=mingw)

F.36 2.26

Released on April 27th, 2007

F.36.1 Release Notes

 Installation on x64 systems is now fully supported with \$PROGRAMFILES64, SetRegView and Library's LIBRARY_X64 option.

F.36.2 Changelog

F.36.2.1 Major Changes

- Added LIBRARY_X64 option for InstallLib and UnInstallLib for installing and uninstalling x64 libraries
- Added \$PROGRAMFILES32, \$PROGRAMFILES64, \$COMMONFILES32 and \$COMMONFILES64

F.36.2.2 Minor Changes

- Added \$EXEPATH and \$EXEFILE
- Added \$(^Language) language string which holds the language name (RFE #1235616)
- Added LIBRARY_IGNORE_VERSION option for InstallLib (patch #1699435)
- Added VXD support for GetDLLVersionLocal on Windows NT4/2000/XP/Vista (patch #1706624)
- Avoid Library warning when UnInstallLib is unused (bug #1692761)
- Fixed error handling of CopyFiles on Windows NT4 (bug #774966)
- Fixed font name encoding broken since 2.24
- Fixed LogicLib duplicate labels across included files
- Fixed preservation of folder attributes with File /a (bug #1699474)
- Improved unused variable warning (bug #1701050)

- Library macros no longer require version information for DLL files on POSIX
- Minor documentation improvements (including RFE #971467)
- Modern UI: added MUI FINISHPAGE CANCEL ENABLED
- Modern UI: added MUI_FINISHPAGE_REBOOTLATER_DEFAULT (RFE #1143843)
- Modern UI: block unsupported languages in the language selection dialog (RFE #1564986)
- Modern UI: disable Cancel button on the finish page (bug #1267491)
- Modern UI: reduce flicker caused by MUI_HEADER_TRANSPARENT_TEXT (patch #1696610)
- Support Quit in show page functions

F.36.2.3 New/Changed Commands

- Added SetRegView to allow access to the x64 registry view
- Installer icon support for MessageBox using MB_USERICON (patch #1682748, RFE #1530388)

F.36.2.4 Utilities and Plug-ins

- InstallOptions 2.47: line breaks support in Link controls (patch #1683186, RFE #1495949), added HLine and VLine controls (patch #1683189)
- MakeNSISW: fixed broken command line parameter handling introduced in the last version (bug #1696534)
- UserInfo: return effective user group on Vista, added GetOriginalAccountType (patch #1687456, bug #1684777)

F.36.2.5 Translations

- Added Afrikaans translation (patch #1699558)
- French corrections (patch #1676101)
- German corrections

F.36.2.6 Build System

- Added SKIPTESTS option
- Avoid a lot of code warnings (bug #1676243)
- Detect wxWidgets instead of assuming its existence on POSIX (bug #1672315)
- Fixed big-endian platform support broken since 2.24

F.37 2.25

Released on March 31st, 2007

F.37.1 Changelog

F.37.1.1 Major Changes

- Added Memento.nsh for easy persistency of user selections across different runs of the installer (RFE #1677624)
- Fixed a bug introduced in 2.24 that allowed the license page to be skipped even with agreement check box or radio buttons (bug #1664428)

F.37.1.2 Minor Changes

- Added Vista manifest to StartMenu.nsi example to avoid backward compatibility mode that moves shortcuts (bug #1664957)
- Both dashes and slashes are supported as switch prefixes on makensis.exe (bug #1661503)
- Delete uninstaller temporary directory on reboot (patch #1660626)
- Distribute Plug-in example with the NSIS package and not only with the source code
- Fixed lossy Unicode conversion of dialog template strings (bug #1662190)
- Fixed Sections.nsh macros support for \$0 as input (bug #1664648)
- Fixed uninstallers support for the /D= command line switch
- Improved logging of WriteReg commands
- Minor documentation updates and fixes (including patch #1662419)

F.37.1.3 New/Changed Commands

- Added bitwise operators support for !define /math (RFE #1669513)
- PAGEEX contains the page type (patch #1644712)

F.37.1.4 Utilities and Plug-ins

- InstallOptions 2.46: Use installer's name for message boxes (bug #1661677)
- MakeNSISW: Use Escape button to close (RFE #1666501)
- Math: Removed mathcrt.lib
- StartMenu: Refuse empty paths (bug #1684751)

F.37.1.5 Translations

Fixed Galician language files (bug #1663795)

F.37.1.6 Build System

- Added APPEND_LIBPATH and APPEND_CPPATH instead of the malfunctioning CPPPATH and LIBPATH
- Added ChangeLog to source code package (patch #1680508)
- Avoid some warnings on VS2005 (patch #1667950)
- Fixed Izma test segfault on POSIX (bug #1666873)
- More strict-aliasing compatibility (bug #1635841)
- NSIS Menu finally built from source

F.38 2.24

Released on February 17th, 2007

F.38.1 Changelog

F.38.1.1 Major Changes

- Library uninstall-reinstall-reboot problems workaround (bug #1097642)
- Minimized number of cases where an error message appears in silent mode
- New compiler predefines for code scope (patch #1644712)

F.38.1.2 Minor Changes

- Added /P command line option for setting process priority of makensis (patch #1638974)
- Added support for 64-bit PE on POSIX to GetDLLVersionLocal (patch #1643633)
- Append last part of InstallDir only to \$INSTDIR on directory pages (bug #1174184)
- Avoid permissions change of output files for LineFind and FileJoin (bug #1631773)
- Fixed erroneous warnings on uninstall sections (bug #1631889)
- Fixed lossy Unicode conversion in resource editor (bug #1083492)
- Ignore invalid preprocessor commands in ignored block or comments
- Made VIAddVersionKey only query language tables instead of creating them and generating warnings (bug #1626504)
- Minor documentation updates and fixes (including bug #1642107, patch #1649187)
- Warn when continuing a comment line using backslash (bug #1554178)

F.38.1.3 New/Changed Commands

 RegDLL and UnregDLL now use LOAD_WITH_ALTERED_SEARCH_PATH, so there's no need to use SetOutPath to set the working directory (bug #1638191)

F.38.1.4 Utilities and Plug-ins

- InstallOptions 2.45: Added FOCUS flag (patch #1634704) and fixed paste with ONLY_NUMBERS flag (bug #1652075)
- MakeNSISW: settings in HKCU instead of HKLM (bug #1411970) and MRU menu accelerators

F.38.1.5 Translations

Added Galician translation (patch #1631765)

F.38.1.6 Build System

- Added a workaround for building on x64 POSIX platforms (bug #1646170)
- Added a workaround for strict-aliasing compatibility (bug #1635841)
- Added compression tests
- Added STRIP, APPEND_CCFLAGS and APPEND_LINKFLAGS build options
- Fixed SCons 0.96.94 compatibility

F.39 2.23

Released on January 13th, 2007

F.39.1 Changelog

F.39.1.1 Minor Changes

- Fixed compiler crash on Mac OS X (patch #1611866)
- Fixed deletion of start menu icons in NSIS installer on Vista (bug #1611251)

F.39.1.2 Utilities and Plug-ins

- Fixed incorrect nsExec message handling in silent mode (bug #1605581)
- Fixed System crash when System::Store is called on an empty private stack (bug #1620178)
- Fixed System crash with parenthesis in filename (bug #1616267)
- Minor VPatch documentation enhancements (patch #1624292)

F.39.1.3 Build System

- Fixed build failures on mingw32 (bug #1610773)
- Fixed build problems with MSTOOLKIT=yes

F.40 2.22

Released on November 27th, 2006

F.40.1 Release Notes

- Some changes have been made to the credit and license files in order to avoid copyrights and license related confusions. This does not change in any way how NSIS should or could be used. NSIS itself is still licensed with the permissive BSD-like zlib license.
- Upgrade is recommended for early adopters of WinVer.nsh, due to a bug in Windows 98 and ME detection.

F.40.2 Changelog

F.40.2.1 Minor Changes

- Fixed context menu not disappearing when moving from instfiles page (bug #1115825)
- Fixed WinVer.nsh's detection of Windows 98 and ME (reported on the forum)
- Fixed WriteUninstaller failure to overwrite read-only uninstallers (bug #1542530)
- Licensed Izma under CPL with a special exception, instead of LGPL
- Minor documentation updates and fixes (including bug #1584618, bug #1589877)
- Updated comments in MUI examples (bug #1595500)

F.40.2.2 New/Changed Commands

Added `highest` option for RequestExecutionLevel

F.40.2.3 Translations

- Added Uzbek translation
- Fixed corruption in Lithuanian (bug #1602673)
- Minor Breton fixes
- Slovenian corrections (patch #1590108)

F.40.2.4 Build System

- Added Microsoft Visual C++ 2005 Express support
- SCons requirement upgraded to 0.96.93

F.41 2.21

Released on October 20th, 2006

F.41.1 Changelog

F.41.1.1 Major Changes

- Added WinVer.nsh for easy Windows version comparisons
- Upgraded to Izma sdk 4.43 for faster compression

F.41.1.2 Minor Changes

- Added Vista support to GetWindowsVersion
- Added x64.nsh including a few simple macros for handling x64 installations
- Fixed a handle leak in Locate
- Minor documentation updates and fixes
- Modern UI 1.76: Added MUI_ABORTWARNING_CANCEL_DEFAULT (RFE #1547844)

F.41.1.3 New/Changed Commands

Added RequestExecutionLevel (RFE #1524709)

F.41.1.4 Translations

- Added Valencian translation (patch #1558822)
- Bulgarian corrections
- Slovenian corrections

F.41.1.5 Build System

• SCons requirement upgraded to 0.96.92

F.42 2.20

Released on September 9th, 2006

F.42.1 Changelog

F.42.1.1 Minor Changes

- Better LogicLib errors (bug #1537976)
- Fixed incomplete plug-in call error messages (bug #1535995)
- Fixed incorrect file timestamp querying on big-endian platforms (bug #1536377)
- Minor documentation updates and fixes

F.42.1.2 Translations

- Danish corrections (bug #1548190)
- Fixed incorrectly encoded Turkish translation (bug #1542765)
- French corrections
- Italian corrections (bug #1546183)
- Spanish corrections
- Swedish corrections (bug #1542680)
- Ukrainian corrections

F.42.1.3 Build System

 Fixed NSIS_CONFIG_CONST_DATA_PATH being ignored on POSIX platforms (bug #1515592)

F.43 2.19

Released on August 6th, 2006

F.43.1 Changelog

F.43.1.1 Minor Changes

- Fixed !system and !packhdr failure with quoted long file names (bug #1509909)
- Fixed build problems on 64bit platforms (bug #1504772)
- Fixed negative total size for data larger than 2GB in script compilation summary (bug #1468852)
- Minor documentation updates and fixes
- Replaced IsDotNETInstalled with a simpler and better version

F.43.1.2 New/Changed Commands

• Made !if compare strings case insensitively, like StrCmp and !ifdef

F.43.1.3 Plug-ins

- NSISdl: fixed random DNS errors (reported in the forums)
- System: fixed a bug that caused stack corruption and stopped the installer from deleting System.dll when a function with no arguments was called (bug #1535005)
- System: fixed wrong return values for functions that return short or char (bug #1535007)

F.43.1.4 Translations

- Added Norwegian Nynorsk translation (patch #1503208)
- Fixed typos in the French translation (bug #1531874)

F.44 2.18

Released on July 1st, 2006

F.44.1 Changelog

F.44.1.1 Minor Changes

- CRC32 implementation used potentially non-32bit types (bug #1504758)
- Fixed errors on multiple inclusion of useful headers
- Fixed GetFileAttributes
- Fixed incorrect text on browse dialog with .onVerifyInstDir (bug #1504297)
- Fixed Library's implementation on POSIX
- Minor documentation updates and fixes

F.44.1.2 Translations

- Added Irish translation (patch #1503639)
- Catalan corrections (bug #1504104)
- Dutch corrections
- Finnish corrections (Mozilla bug #341643)
- Fixed Serbian grammar and typos
- Italian corrections (Mozilla bug #340450)
- Polish corrections (Mozilla bug #224532)
- Portuguese (Brazil) corrections (Mozilla bug #340885)
- Romanian corrections (Mozilla bug #340645)
- Thai corrections
- Turkish corrections (Mozilla bug #340511)

F.45 2.17

Released on May 19th, 2006

F.45.1 Changelog

F.45.1.1 Minor Changes

- Fixed build failures and segfaults on PowerPC Mac OS X (bug #1474597, bug #1481044)
- Fixed Library's TLB version interpretation (bug #1471341)
- Fixed possible stack corruption when using TypeLib.dll on an invalid TLB
- Fixed RMDir deletion failure of read-only folders (bug #1481664)
- MakeNSISW: Fixed toolbar compressor selection menu, broken in the previous version (bug #1466486)
- Minor documentation improvements and fixes (including bug #1469306, bug #1491616)

F.45.1.2 Translations

- Added missing Basque and Welsh files (bug #1469471)
- Brazilian Portuguese updates
- Bulgarian fixes
- Fixed a typo in Hebrew translation (bug #1474587)
- Icelandic fixes and improvements

F.46 2.16

Released on April 7th, 2006

F.46.1 Release Notes

- The script compiler, makensis, builds and works on big-endian platforms. This change enlarges the portability range of NSIS to theoretically every POSIX platform. Please report any incompatibility with specific platforms or build-tools.
- The internal changes made to support big-endian platforms also pave the road to x64 installers. There is now a central function which writes data to disk. This function currently only converts the endianity of integers, but it can be changed to selectively write 64-bit integers. Hopefully, there'll soon be a simple method of compiling a script to both x86 and x64 installers.
- Changing Source/exehead/fileform.h to alter the internal structure of installers is no longer enough. The compiler has its own definitions of the structures which must also be changed in Source/fileform.cpp. In the future, fileform.cpp should be automatically generated from fileform.h, but for now, the synchronization must be done manually.

F.46.2 Changelog

F.46.2.1 Major Changes

- Big-endian platforms are now fully supported by makensis
- Library now available on non-Windows platforms as well, although it requires the installed DLL to have version information
- MakeNSISW 2.1: added "Cancel compilation" menu item

F.46.2.2 New/Changed Commands

Added /utcdate switch to !define for UTC dates (RFE #1459210)

F.46.2.3 Minor Changes

- Added an optional timestamp in the log -NSIS_CONFIG_LOG_TIMESTAMP build setting (RFE #1460586)
- Added NSIS_WIN32_MAKENSIS define, defined only when compiling on Windows
- Exported validate_filename to plug-ins
- Fixed a crash in CResourceEditor when adding resources to a PE that already contains named resources
- Fixed a small resource leak in the TypeLib::GetLibVersion plug-in function
- Fixed CResourceEditor input sanity checks
- Fixed incorrect FileOpen input validation (bug #1459789)
- Fixed Library failure with DLLs marked as read-only
- Fixed Izma's POSIX implemention thread synchronization issues and resource leaks
- Fixed makensis self-path detection on non-Windows platforms (NSIS_CONFIG_CONST_DATA_PATH=no)
- Fixed replace_icon and generate_uninstall_icon_data icon validation
- Made external CHM links safer to script exceptions (bug #1449879)
- Minor documentation improvements and fixes (including bug #1077439, bug #1448374, RFE #1464446)
- Modern UI 1.75: added show function for the start menu page (RFE #1448176), added MUI_HEADER_TRANSPARENT_TEXT for transparent header texts (RFE #1447766)
- NSISdl: better header detection for better compatibility with proxies like WinProxy (bug #1445735), fail if no headers are sent, faster downloads (patch #1465378)
- StartMenu: validate user input (bug #1440636)

F.46.2.4 Translations

- Added Basque translation
- Minor Slovenian fixes

F.46.2.5 Build System

- Added code tests for CResourceEditor and CDialogTemplate
- Automatically pass build settings to script, eliminating the need to edit build.cpp for new settings
- Fixed \$PREFIX expansion during installation on POSIX platforms (bug #1456943)
- Fixed CHM dependencies
- Fixed compatibility issue with EclipseNSIS (version wasn't prefixed with 'v')
- Fixed GCC 4.1 compatibility (patch #1456861)
- Test for -WI,-Map availability

F.47 2.15

Released on March 4th, 2006

F.47.1 Changelog

F.47.1.1 New/Changed Commands

- Added !if for more complex compile-time flow control (patch #1412982)
- Added /math switch to !define for simple compile-time mathematical operations (patch #1372561)

F.47.1.2 Minor Changes

- Added more replacement options to WordReplace
- Added NSIS_CONFIG_LOG_STDOUT configuration option for logging to stdout
- Added path translation for !addincludedir (bug #1431958)
- ConfigRead now sets the error flag, if the entry wasn't found
- Documented Nop
- Edit box in the installation directory selection page is now always LTR, even for RTL languages
- Fixed improper iterator usage in dir reader (bug #1431593)
- Fixed MessageBox MB_TOPMOST not showing up, if used as the first sections' instruction (bug #1400995)
- Fixed RMDir skipping files with names starting with two dots (bug #1420657)
- GetOptions now sets the error flag, if the option wasn't found
- Made !include stop searching the include directories after a match is found (bug #1441877)
- Made header functions use /NOUNLOAD for faster plug-in calls
- Minor documentation improvements and fixes (including bug #1420352, bug #1432423)

- More informative Icon error messages (bug #1174742)
- New case-sensitive functions in headers: WordFindS, WordFind2XS, WordFind3XS, WordReplaceS, WordAddS, WordInsertS, StrFilterS, TextCompareS, ConfigReadS, ConfigWriteS, GetOptionsS

F.47.1.3 Translations

- Added proper language identifier for Breton
- Breton translation improvements
- Bulgarian translation improvements
- Fixed Czech grammar mistakes (patch #1427189)
- Fixed Italian typo (patch #1416988)
- Updated Mongolian

F.47.1.4 Build System

- Fixed __BIG_ENDIAN__ definition
- Fixed improper handling of paths passed to the compiler (bug #1434215)
- Improved GCC 4.2 compatibility (patch #1434174)
- Moved NSIS_VARS_SECTION from config.h to scons
- Write all configuration to sconf.h instead of passing it on the command line

F.48 2.14

Released on January 24th, 2006

F.48.1 Release Notes

• This release fixes a critical bug that caused installers using plug-ins to fail loading on Windows 9x and NT. Upgrading from 2.13 is highly recommended.

F.48.2 Changelog

F.48.2.1 Major Changes

 Fixed a bug that corrupted \$TEMP under Windows 9x and NT and caused InitPluginsDir to fail (bug #1412159)

F.48.2.2 Minor Changes

• Fixed a possible buffer overflow caused by long values of \$0, when using large NSIS_MAX_STRLEN builds (above 4096)

F.49 2.13

Released on January 21st, 2006

F.49.1 Release Notes

• Language detection has changed. The default language is now set to the user's user-interface language instead of the locale language, as suggested by MSDN. To restore the old behavior, use System::Call "kernel32::GetUserDefaultLangID()i.a" in .onlnit.

F.49.2 Changelog

F.49.2.1 Major Changes

 Default \$LANGUAGE value is now based on user's UI language instead of locale (bug #1324734)

F.49.2.2 New/Changed Commands

 Added StrCmpS for case sensitive string comparison (patch #1381929)

F.49.2.3 Minor Changes

- Added system time support to GetTime
- Fixed components page checkbox redraw problem under Windows 95 (bug #1397031)
- Fixed constant maximum string length in FileRead (changed from 1024 to NSIS_MAX_STRLEN)
- Fixed empty \$INSTDIR value in directory page show callback function (bug #1209843)
- Fixed relative jumps ignoring File /nonfatal (bug #1299100)
- Fixed typo in NSIS Menu (bug #1387748)

- Issue a warning when BrandingText /TRIM* actually expands the label (bug #1362443)
- Made RMDir /r remove Unicode paths as well using short names (bug #1378785)
- Minor documentation improvements and fixes

F.49.2.4 Utilities and Plug-ins

- Fixed System plug-in documentation of callback functions (bug #1403608)
- Fixed System plug-in heap corruption (bug #1403601)

F.49.2.5 Translations

Mongolian translation improvements

F.49.2.6 Build System

• Better installation under POSIX with proper \${NSISDIR} detection and appropriate directory paths (/usr/bin, /usr/share/doc, etc.)

F.50 2.12

Released on December 17th, 2005

F.50.1 Changelog

F.50.1.1 Major Changes

- Fixed compile-time assertion failure for big uninstallers (bug #1380447)
- Tutorial enhancements (including bug #1366431)

F.50.1.2 New/Changed Commands

Added /NONFATAL switch to !include (patch #1372048)

F.50.1.3 Minor Changes

- Clearer error message for double label definition (patch #1374675)
- Fixed browsed network root directory not being accepted (bug #1331292)
- Fixed incorrect CompletedText parsing (bug #1349810)
- Fixed sdbarker_tiny.exe's compatibility with NSIS_CONFIG_LOG (bug #1365869)
- Implemented nicer registry commands log (patch #1340255)
- Minor documentation improvements and fixes (including patch #1355653, bug #1349810)
- Multi-line comments are no longer ignored inside !ifdef'd block

F.50.1.4 Utilities and Plug-ins

- Added drag & drop support for zip2exe
- Better drag & drop error handling in MakeNSISw

• Fixed Math plug-in array issues (bug #1235875)

F.50.1.5 Translations

- Added browse button Danish translation
- Breton translation fixes and improvements
- Fixed finish button Swedish translation

F.50.1.6 Build System

- Fixed endianess detection problem on POSIX platforms (bug #1370179)
- Generated installers use nsis-VERSION-setup.exe template

F.51 2.11

Released on November 12th, 2005

F.51.1 Release Notes

• Rebuilding existing installers that use SetOverwrite ifdiff, might falsely overwrite files once

F.51.2 Changelog

F.51.2.1 Major Changes

- Added a workaround for a bug that prevented detection of some special folders (e.g. \$DESKTOP for all users) on Windows 9x (bug #1008632)
- Fixed a crash caused by copying very long details to clipboard in the installation log page (bug #1314004)

F.51.2.2 New/Changed Commands

- Added !tempfile, !delfile and !appendfile for cross-platform handling of text files during compilation
- Fixed ChangeUI input handling (patch #1348473)
- SectionIn RO is no longer case sensitive

F.51.2.3 Minor Changes

- Added support URL to the NSIS package entry in the Add/Remove control panel (RFE #1349867)
- Fixed comment handling (patch #1324898)
- Fixed duplicate RegTool test in Library
- Fixed invalid language selection according to \$LANGUAGE in .onInit when only primary language match is found (bug #1328629)

- Fixed missing RegTool error after install-reboot-install-reboot sequence with Library
- Fixed portability issues with LogicLib (bug #1320297, patch #1248336)
- Fixed SetOverwrite ifdiff always overwriting on FAT file system (bug #1338423)
- Fixed wrong size of red.bmp check box image
- Larger browse button for localized texts (bug #1314682)
- Minor documentation improvements and fixes (including bug #1349810)
- Sort language names in language selection dialog
- Use the temporary directory for Library temporary files, instead of the possibly write-protected installation directory

F.51.2.4 Utilities and Plug-ins

- Added /OEM switch for OEM to ANSI conversion in nsExec (patch #1346737)
- Added /PROXY switch for manual proxy configuration in NSISdI (patch #1334166)
- Added solid compression check box to zip2exe (patch #1334155)
- Added stdin in MakeNSISw to allow xcopy to run
- Added timestamp handling to zip2exe (bug #1349853)
- Fixed input validation of nsExec
- InstallOptions 2.44: Added HWND and HWND2 entries to the INI file to avoid messy calculations of the correct control id

F.51.2.5 Translations

• Slovenian translation fixes and improvements

F.51.2.6 Build System

 Added a workaround for linking errors caused by a bad library included in recent Platform SDK versions

- Check for and use -pthread linker flag
- Improved FreeBSD portability
- Nicer error message for SCons version older than 0.96.90

F.52 2.10

Released on October 4th, 2005

F.52.1 Changelog

F.52.1.1 Major Changes

- Added auto completion to the directory page
- Fixed a bug, introduced in 2.09, that caused AllowRootDirInstall to fail
- Fixed a thread leak in the POSIX implementation of LZMA
- VPatch 3.1: MD5 checksums, better performance, and some bug fixes (including bug #1219806)

F.52.1.2 Minor Changes

- Added rounding of required and available size on the directory page (1.59 => 1.6 instead of 1.5)
- Added WS_EX_LEFTSCROLLBAR style in RTL mode (bug #1283528)
- Fixed alteration of the working directory by FileRequest in InstallOptions (bug #1287731)
- Fixed bad mnemonic key in MakeNSISw menu (bug #1288159)
- Fixed negative size values showing up after the decimal point in the directory page for very big sizes
- Minor documentation improvements and fixes
- Modern UI 1.74: Fixed compile error when checkboxes are used on multiple finish pages

F.52.1.3 Translations

 Fixed a bug in Slovenian translation that caused lots of missing language string warnings

- Fixed typos in Serbian translation
- Fixed typos in Thai translation
- Updated Simple Chinese translation

F.52.1.4 Build System

- Automatic fix for VC6 SP6 compile error
- Moved most of the configuration from config.h to the build system
- Removed all optimizations and symbol stripping in debug mode
- SCons requirement updated to 0.96.91
- VPatch can be built on POSIX as well

F.53 2.09

Released on August 26th, 2005

F.53.1 Changelog

F.53.1.1 Major Changes

- Fixed a bug introduced in 2.08, that prevented uninstallers from deleting \$INSTDIR
- Fixed a bug that caused "C:\" to be considered a valid installation directory, even without AllowRootDirInstall

F.53.1.2 New/Changed Commands

 Var can now be used in sections and functions, but only with the /GLOBAL flag

F.53.1.3 Minor Changes

- Applied patch #1248335 for greater portability of examples
- FileFunc: workaround for GetLongPathName which is not available on Windows 95 (Instructor)
- Fixed an unused label warning in Library
- Fixed duplicate label errors in LogicLib and Library when used in different files (bug #1243865)
- Modern UI 1.73: fixed checkboxes on uninstaller finish page or multiple finish pages
- WordFunc: fixed incorrect replacement of first word in WordReplace, under some conditions (Instructor)

F.53.1.4 Translations

- Renamed Malaysian to Malay
- Update Thai translation to the latest version
- Updated Kurdish MUI translation to the latest version

F.53.1.5 Build System

• Fixed a number of build problems on POSIX platforms which caused plug-ins to malfunction and installers to crash

F.54 2.08

Released on July 23rd, 2005

F.54.1 Release Notes

• The Archive has been replaced with a Wiki. The new Wiki allows everyone to edit all pages so there's no longer need to hunt for the original author. It also allows everyone to upload plug-ins and not just administrators. And as if that's not enough, it looks better and provides easier and more feature-rich syntax.

F.54.2 Changelog

F.54.2.1 Major Changes

- Added Instructor's header files of useful functions
- Library improvements: ordered registration after reboot, smaller RegTool and separate process for each registration to avoid conflicts (thanks stb)

F.54.2.2 Minor Changes

- Added Bosnian and Kurdish translations
- Added per-user nsisconf.nsh file in %APPDATA% or \$HOME, depending on the platform (patch #1223041)
- Documentation improvements and fixes (including bug #1202495, bug #1227610, bug #1238686, patch #1225167, RFE #1240601)
- Fixed _?= being ignored, if the uninstaller path is not quoted and is separated with only space from ?=
- Fixed Library failing on paths with spaces (bug #1234283)
- Fixed UpgradeDLL compilation error (bug #1230336)
- Improved French, Ukrainian and Luxembourgish translation
- Made RMDir set the error flag, if passed an invalid directory path

(bug #1227553)

- Made uninstallers copy themselves into a subdirectory of the temporary directory to avoid DLLs left in the temporary directory from being loaded by the uninstaller (patch #1214319)
- Missing LangString warning now uses the language name, if possible
- zip2exe: fixed restriction of extraction path length and updated to zlib 1.2.3 (bug #1226381)

F.54.2.3 Build System

- Added linker script to assure correct order of sections when building using GNU tools
- Added test target
- Uls are now built from source

F.55 2.07

Released on June 20th, 2005

F.55.1 Release Notes

- NSIS Update was removed and the update check was moved back to MakeNSISw (Help -> NSIS Update). Use the nightly builds to get the latest compiled binaries from CVS.
- Solid compression is no longer the default for bzip2 and lzma, use SetCompressor /SOLID bzip2 Or SetCompressor /SOLID lzma for solid compression.
- Source code is no longer supplied with the installer package. Source code can be downloaded from CVS or from the download page.
- The directory structure of the installation has been changed. All documentation is now in the Docs folder and all examples are in the Examples folder.

F.55.2 Changelog

F.55.2.1 Major Changes

- Added /SOLID switch for SetCompressor
- New build system (see appendix G)
- Stubs (exeheads) are no longer compiled into makensis.exe, but kept in the Stubs folder

F.55.2.2 New/Changed Commands

- EnumRegValue sets the error flag, if the enumeration index is out of range (bug #1178756)
- ExpandEnvStrings now sets the error flag as the documentation states
- Made File /oname throw an error if the first character is a quote

F.55.2.3 Minor Changes

- Added \$LOCALAPPDATA (RFE #1172123)
- Changed MUI dialogs charset from ANSI_CHARSET to DEFAULT CHARSET (bug #1193736, bug #1201712)
- Fixed \$INSTDIR changing, even if user clicked Cancel in browse dialog
- Fixed a crash in makensis when using Windowlcon off (bug #1123353)
- Fixed Call :label in uninstall code
- Fixed compilation error without NSIS CONFIG COMPONENTPAGE
- Fixed compilation errors on various platforms (including patch #1179116, patch #1193692)
- Fixed CreateDirectory setting the error flag for directories with two consecutive backslashes (bug #1119442)
- Fixed InstallLib setting the error flag for new shared libraries (bug #1181951)
- Fixed letters case in examples (patch #1184571)
- Fixed logging state being reset after the directory page (bug #1168711)
- Fixed overlapping icon in welcome page (bug #1221772)
- Fixed RMDir /REBOOTOK setting the reboot flag when trying to delete non-existent directory (bug #1073792)
- Minor documentation fixes and improvements (including bug #1220940)
- Removed NSIS Update, MakeNSISw checks for new versions again

F.55.2.4 Include Files

- More messages in WinMessages.nsh (Shengalts Aleksander)
- Some more useful macros in Colors.nsh (Joel)

F.55.2.5 Utilities and Plug-ins

• Fixed halibut segfaults on PowerPC (patch #1180886)

- Math: Fixed memory leak (brainsucker)
- zip2exe: upgraded to zlib 1.2.2

F.55.2.6 Translations

- Added Malaysian translation (Azwa)
- Added Welsh translation (Rhoslyn Prys)
- Fixed Breton NLF file (credits weren't commented)
- Fixed duplicate accelerator in French translation (bug #1168652)
- Improved French translation (bug #1220277)
- Updated Russian translation (Dmitry)
- Updated translation credits (patch #1223362)

F.56 2.06

Released on March 19th, 2005

F.56.1 Changelog

F.56.1.1 Major Changes

- Fixed a bug, introduced in 2.05, that made components selection not function properly if the first section in the script was a section group
- Fixed a bug, introduced in 2.05, that corrupted the state of section groups inside section groups (bug #1155836)

F.56.1.2 New/Changed Commands

- Added MB RTLREADING style to MessageBox (RFE #1159701)
- All registry instructions now accept SHCTX which is replaced with HKLM or HKCU at runtime according to SetShellVarContext (RFE #1124901)

F.56.1.3 Minor Changes

- Added Belarusian and Icelandic language files
- Added components.c to the Makefile (patch #1123154)
- Added missing ClearErrors in GetWindowsVersion (bug #1155588)
- Fixed a bug in install.sh that made it try to install a non existing directory
- Fixed a bug in the Russian language file distributed with 2.05
- Fixed a bug that caused SetFont to not function properly with RTL languages on Windows 9x (bug #1159700)
- Fixed a bug, introduced in 2.05, that caused problems with InstType /COMPONENTSONLYONCUSTOM (bug #1155060)
- Fixed a crash caused by compiling large uninstallers (bug #1144763)
- Fixed compile error without NSIS_CONFIG_CRC_SUPPORT

- Fixed compile errors of MinGW on POSIX for source code under Contrib (patch #1164307)
- Fixed plug-ins directory initialization for extremely restricted guest users with no directory listing access to the temporary directory
- Made install.sh not use \$(tempfile) and install files under /usr/share instead of /lib (patch #1120399)
- Minor documentation updates and fixes
- Updated Slovenian, Serbian and Serbian Latin language files

F.56.1.4 Utilities and Plug-ins

- AdvSplash: Fixed a small memory leak (Thanks Takhir)
- StartMenu: Added support for SetCtlColors (RFE #711900)

F.57 2.05

Released on February 4th, 2005

F.57.1 Release Notes

 A lot of changes were made to the behavior of sections/components and related instructions. The changes were thoroughly tested, but may still cause minor incompatibilities with old scripts. Please let us know if your script no longer functions as expected with these changes.

F.57.2 Changelog

F.57.2.1 New/Changed Commands

- Renamed SubSection and SubSectionEnd to SectionGroup and SectionGroupEnd
- SectionSetFlags, when used on a section group, toggles its children too
- SectionSetText works better with variables (parses immediately and treats an empty variable as an empty string)
- SetCurInstType works without the components page

F.57.2.2 Minor Changes

- Added Breton translation
- AdvSplash: Fixed double delay time in some cases
- Documentation fixes and enhancements (including patch #1098454)
- Fixed case sensitive name comparison with File /x, when not using wildcards
- Fixed extraction status ending prior to 100%
- Fixed negative values of available space in the directory page (bug #1114876)

- Fixed system.nsi example crash (bug #1102255)
- Fixed unused label warnings with Library macros
- Improved some error messages
- InstallOptions 2.42: Added an image displaying example, added TRANSPARENT flag for BITMAP fields (RFE #1079715 funded by Chris Morgan)
- Invisible sections weren't always affected by a change of the installation type (bug #1045722)
- Language file fixes
- Made pages refresh after an aborted leave function
- Partially selected section groups can now be toggled

F.58 2.04

Released on January 7th, 2005

F.58.1 Changelog

F.58.1.1 Major Changes

- Fixed \$0 changed by File /r
- Fixed empty directory not included by File /r
- Fixed invalid directory creation by File /r on POSIX
- Fixed Unicode conversion problems on POSIX

F.58.1.2 Minor Changes

- Added new test for dir reader::matches
- Fixed LibraryLocal failure with UNC paths
- Fixed VC 7.1 and GCC 3.4 compile errors
- Language file fixes

F.59 2.03

Released on December 3rd, 2004

F.59.1 Release Notes

- If you're using File /r folder in your script, it's recommended you replace it with File /r folder* for faster compilation and so no other folders named *folder* will be included
- Command line switches prefix on POSIX has changed to a dash
- The plug-in API change is backward compatible

F.59.2 Changelog

F.59.2.1 Major Changes

- Added another parameter to plug-in functions with a pointer to exec flags and ExecuteCodeSegment
- Fixed handling of absolute paths on POSIX
- Made RegDLL load and unload a DLL exactly once to fix crashes with COM DLLs registration

F.59.2.2 New/Changed Commands

- Added /date switch to !define for definition of date and time constants
- Added /x switch for File and ReserveFile to exclude files and directories
- Made File's /r switch always search in subdirectories, even if the given path points to an existing directory

F.59.2.3 Minor Changes

Added LIBRARY_SHELL_EXTENSION and LIBRARY_COM to

Library

- Added missing #include
- Added NSIS_LZMA_COMPRESS_WHOLE to the script define list
- Documentation fixes and improvements
- Fixed an access violation in case of a script jump beyond the last entry
- Fixed keyboard navigation in the instfiles page while installing
- Fixed unpacking window not showing when BGGradient is used
- Fixed unused label warning in Library.nsh
- Language file fixes
- Modern UI 1.72: Fixed state of Cancel button on Finish page when used in installer and uninstaller, added a string for NSIS CONFIG COMPONENTPAGE ALTERNATIVE
- nsisconf.nsh was not always parsed when compiling using MakeNSISw

F.60 2.02

Released on October 23rd, 2004

F.60.1 Changelog

F.60.1.1 Major Changes

- Added lots of small usage examples to the documentation
- Made relative jumps work with instructions that add multiple entries
- Made the datablock optimizer much faster
- Made the installer deny reboots while running (WM QUERYENDSESSION)
- Made the Reboot command first quit and then reboot so everything is always cleaned-up

F.60.1.2 New/Changed Commands

- Added << and >> to IntOp
- Added GetErrorLevel and SetErrorLevel
- Fixed CopyFiles usage checking problem
- Made Reboot call .onRebootFailed and quit on failure instead of setting the error flag

F.60.1.3 Minor Changes

- Added Albanian language files
- Added NSIS_CONFIG_COMPONENTPAGE_ALTERNATIVE configuration option which makes components only be toggled when the user clicks on the checkbox and makes .onMouseOverSection only be called when the user selects a component
- Added some CppUnit tests
- Both _?= and /D= now require a space before them so they can safely remove the space from \$CMDLINE

- Fixed a bug that caused the uninstaller CRC-check to fail if !packhdr created a non-512-bytes-aligned exehead
- Fixed a compiler crash caused by using GetCurrentAddress
- Fixed errors when using 'File /r .' on Windows
- Fixed gcc 3.4 compile errors
- Fixed inconsistencies between error levels and the documentation
- Fixed some minor memory leaks
- Language file fixes
- Makefile improvements
- Modern UI 1.71: Made the selected language only be saved if the installation was successful
- Some code refactoring
- StrFunc: Rewrote StrSort and fixed StrStrAdv. See the readme for a complete changelog

F.60.1.4 Utilities and Plug-ins

Banner: Fixed some cases where the banner would not show on the foreground

F.61 2.01

Released on September 24th, 2004

F.61.1 Release Notes

- See Compiling NSIS Sources for information about compiling makensis on POSIX platforms
- NSIS doesn't create installers for Linux/Mac OS X etc., but you can compile Windows installers on these platforms
- UpgradeDLL is still provided in UpgradeDLL.nsh for backwards compatibility

F.61.2 Changelog

F.61.2.1 Major Changes

- Compiler (makensis) supports POSIX platforms (Linux, *BSD, Mac OS X, etc.)
- New system for DLL/TLB library setup

F.61.2.2 New/Changed Commands

- Added BGFont that allows setting the background text font
- Added SW_HIDE to ExecShell's accepted show modes
- RMDir can now be used with both /r and /REBOOTOK at the same time
- Extended maximum binary data for WriteRegBin to 3 * NSIS_MAX_STRLEN
- Added !execute

F.61.2.3 Minor Changes

• LZMA exehead is now 34KB and should also decompress faster

- Windows 95 (OSR2)/98/ME no longer see drive free space capped to 2GB
- Modern User Interface: New orange theme by MoNKi
- Speedup background gradient painting
- LogicLib 2.5: Added AndIf, AndUnless, OrIf, OrUnless. Avoids unused variable warnings by requiring !defines before using certain features.
- StrFunc: Added support for uninstaller, some fixes and improvements. See the readme for a complete changelog
- The plug-ins folder is properly removed when the system is rebooted
- SetOutPath "-" works again
- Fixed a bug which made plug-ins that didn't have lower case extension not be found
- Fixed a bug that caused directories with drives that had their current directory set to an invalid directory to not be accepted in the directory selection page
- Fixed a crash in makensis caused by defining a macro in a file included by another macro
- makensis shows meaningful errors for compression errors instead of just magic numbers
- Fixed a bug with AllowRootDirInstall used along with InstallDirRegKey that caused the directory in the registry to be ignored
- Fixed FileRead setting the error flag when a null character is the first character it reads
- Fixed a bug which caused the background gradient to paint slowly
- File paths relative to the root folder work again
- Added Colors.nsh
- Made Times New Roman default font for the background text because it should always have support for the locale's language
- Fixed compilation of NSISdl under VS.NET
- SetCtlColors /BRANDING wasn't working right, if just one of the background or text color were specified
- Language file fixes and improvements

F.61.2.4 Utilities and Plug-ins

- System: New, hopefully more informative, documentation; fixed some bugs
- Banner: Added getWindow to allow greater control over the banner window
- InstallOptions 2.41: Bitmaps are now automatically centered, fixed a bug which prevented enabling the next button from the leave function of InstallOptions pages, fixed a rare freeze
- Zip2Exe 0.32: Fixed codepage problems
- nsExec: Always create a valid input handle, fixed a problem when called from a path with spaces
- VPatch: Close all open file handles when one of them fail to open
- NSISdl: Added /NOIEPROXY (based on memph's code)

F.62 2.0

Released on February 7th, 2004

F.62.1 Release Notes

• Finnish, Bulgarian and Thai language files are outdated and do not contain all needed translations

F.62.2 Changelog

F.62.2.1 Changes from 1.98

- Multiple languages in one installer support (with RTL support)
- Easier plug-in system
- Modern User Interface (optional)
- LZMA compression which provides installers 20% smaller than bzip2
- Easier paging system (no more .onNextPage, .onPrevPage)
- Components are presented as a tree which allows sub components
- User variables (\$VARNAME)
- Icon and UninstallIcon support any color depth and sizes, not just 32x32x16
- CheckBitmap supports any color depth
- Improved large files handling
- License data can be RTF
- CHM documentation for easy browsing
- Better silent support
- New include files for easier scripting: LogicLib, StrFunc and Sections
- Improved plug-ins: InstallOptions, NSISdl and Splash
- New plug-ins: AdvSplash, Banner, BgImage, Dialer, LangDLL, Math, nsExec, StartMenu, System, UserInfo and VPatch
- New and improved utilities: MakeNSISw, NSIS Menu (NSIS.exe), NSIS Update (Bin\NSIS Update.exe) and zip2exe (Bin\zip2exe.exe)
- New commands: !addplugindir, !echo, !ifmacrodef, !ifmacrondef, !verbose, AddBrandingImage, AllowSkipFiles, ChangeUI,

CheckBitmap, CreateFont, DirVar, DirVerify, EnableWindow, FileBufSize, FlushINI, GetCurInstType, GetDlgItem, GetInstDirError, IfAbort, IfSilent, InitPluginsDir, InstTypeGetText, InstTypeSetText, LangString, LicenseBkColor, LicenseForceSelection, LicenseLangString, LoadLanguageFile, LockWindow, Page, PageEx, ReserveFile, SectionGetInstTypes, SectionGetSize, SectionSetInstTypes, SectionSetSize, SetBrandingImage, SetCompressionLevel, SetCompressor, SetCompressorDictSize, SetCtlColors, SetCurInstType, SetPluginUnload, SetSilent, ShowWindow, SubSection (replaced by SectionGroup), SubSectionEnd (replaced by SectionGroupEnd), Var, VIAddVersionKey, VIProductVersion and XPStyle

- Removed commands: DirShow, DisabledBitmap, EnabledBitmap and SectionDivider
- All this and a smaller overhead! =)
- A lot more... See below for more information.

F.62.2.2 Changes from RC4

- Command line parser ignored any switches after tokens that are not switches
- Last part of the path specified in InstallDir was appended even if selected folder name was the same
- Modern UI 1.70: Improved documentation, new Init custom function for Welcome and Finish page
- Added StrFunc by deguix
- Updated and fixed language files
- Updated, fixed and improved documentation

F.63 2.0 Release Candidate 4

Released on February 2nd, 2004

F.63.1 Changelog

F.63.1.1 Major Changes

• Fixed MBCS mishandling in exehead, InstallOptions and nsExec

F.63.1.2 Minor Changes

- Language files updates and fixes
- Fixed logging (NSIS_CONFIG_LOG)
- Fixed compile errors caused by removing some config.h options
- NSIS compiles without PSDK again
- Documentation fixes

F.63.1.3 Utilities and Plug-ins

- NSISdl: All potential and rare crashes should be completely fixed now
- InstallOptions: Fixed a rare crash related to ListItems and a small memory leak when using ValidateText

F.64 2.0 Release Candidate 3

Released on January 26th, 2004

F.64.1 Changelog

F.64.1.1 Minor Changes

- Fixed some bugs with LZMA/bzip2 and non-solid compression
- Fixed a bug that caused escaping not to be ignored on extended !define lines
- Language files updates and fixes
- Fixed installer crash when certain language strings were not referenced in all languages
- Some new and improved graphics
- Made CreateShortcut case insensitive when parsing hot key
- Some documentation improvements and fix-ups
- Modern UI 1.69: Made all uninstaller pages work without installer pages and fixed top text on uninstaller license page

F.64.1.2 Utilities and Plug-ins

- MakeNSISW: Fixed resize bug, fixed some UI issues in the settings dialog
- NSIS Update: Improved UI
- InstallOptions: Fixed minor problems with the new NOTIFY flag

F.65 2.0 Release Candidate 2

Released on January 5th, 2004

F.65.1 Changelog

F.65.1.1 Minor Changes

- Set focus to the main control in each page to ease keyboard control (plug-ins too)
- LangStrings and user variables limit is now 16383 and is enforced
- Fixed control colors set using SetCtlColors not changing when the system colors were changed
- Log window now responds to the context menu key
- Fixed a bug which caused beeping when the space key is hit on the components tree
- Added code to prevent weird usage of WM_COMMAND which can cause weird behavior
- Fixed compile errors when NSIS_CONFIG_COMPONENTPAGE, NSIS_CONFIG_PLUGIN_SUPPORT or NSIS_SUPPORT_HWNDS are not defined
- More language files updated to the latest version
- Fixed CHM's script errors
- Documentation improvements

F.65.1.2 Utilities and Plug-ins

- InstallOptions 2.4: NOTIFY for link, drop list and list box; UI fixes and improvements. See IO's change log for a complete list
- MakeNSISW: Improved user interface, added support for named symbol sets and fixed best compressor selector

F.66 2.0 Release Candidate 1

Released on December 27th, 2003

F.66.1 Release Notes

- Trying the LZMA compression method for your installer is recommended. It often gives a 20% better compression ratio.
- Modern UI 1.68: The setting to change the title size on the Welcome page and Finish page has been changed

F.66.2 Changelog

F.66.2.1 Major Changes

- LZMA compression support. Added SetCompresssorDictSize.
- Modern UI 1.68: New settings for extra space for title and text on Welcome page and Finish page, improved handling of verbose settings, language file string for uninstaller reboot information, more
- Shell folders are now detected using shell API functions and not the registry. This is the recommended method by Microsoft. New constants have been added in the process (\$FONTS, \$SENDTO and more...). See section 4.2.3 for more details.
- Added LogicLib to distribution (easier conditional execution etc.)
- Added DirVerify and GetInstDirError to allow custom error checking when the installation directory is invalid or the drive does not have enough space
- Support for reading environmental variables at compile time: \$\%envVarName\%
- Added /SD parameter for MessageBox. Allows to set default for silent installers.
- New conditional compilation options: !ifmacrodef and !ifmacrondef.
- New sections macros for mutually exclusive section selection
- Improved RTL support by adding WS_EX_RTLREADING wherever possible

F.66.2.2 Utilities and Plug-ins

- InstallOptions 2.3: Added new control type "Button", added new flag "NOTIFY", added new flag "NOWORDWRAP" for multi-line text boxes, reduced size down to 12K
- MakeNSISW: More options to set the compression method
- Zip2Exe 0.31: Fixed compression setting, LZMA compression support
- VPatch 2.1: Better error handling and exit code for GenPat

F.67 2.0 Beta 4

Released on November 19th, 2003

F.67.1 Release Notes

- The /LANG parameter is no longer available for any text setting instructions. You must use a LangString if you want to make a certain text multilingual. This means you can also set one text for all languages. To make the license data multilingual you should use LicenseLangString.
- Modern UI 1.67: Because of the new syntax for pages, renaming of settings and variable names etc., you have to make some changes to your scripts, see the Modern UI Readme for details.
- Custom Pages now have a leave function. As this parameter is placed before the caption you have to add another "" empty string for the title to work.
- .onSelChange is no longer called when the components page is created.
- The icons folder has been renamed to Graphics and was reorganized. If you were using any files from Contrib\lcons in your script, you should update it to point to the new image or icon path.
- UpgradeDLL has been changed, it's highly recommend that you include the new version in your script using !include "UpgradeDLL.nsh" instead of the old one.

F.67.2 Changelog

F.67.2.1 Major Changes

- CHM documentation searchable and comes with an index
- User variables (\$VARNAME) that can be declared with the Var command
- Support for all pages in both installer and uninstaller. Components, directory and license pages can be used for the uninstaller

- Improved large files handling (way lower requirements to compile a 2GB installer now)
- Full support for RTL languages, including support for LTR and RTL languages in one installer
- Modern UI 1.67: New system & syntax for pages, settings (support for multiple pages of the same type, page specific settings, more customization options etc.), Welcome/Finish pages for uninstaller, LicenseForceSelection support, new options for Finish page / language selection dialog, fixes, more
- Extraction progress
- No more unprocessed strings, variables can be used everywhere
- Leave function for custom pages: Input on InstallOptions pages can be validated using script code
- Interface improvements: Better ClearType support, no more flickering
- OnMouseOver text is only displayed when mouse over section
- NLF language files (v6): language specific fonts, RTL and more strings
- Inner LangStrings can be used in the script
- No more /LANG, only LangStrings easier to set one string to all languages
- LangStrings are no longer installer/uninstaller specific (no un.)

F.67.2.2 New/Changed Commands

- Added PageEx, PageCallbacks and DirVar it's now a lot easier to add the same page type twice
- Added FileBufSize
- Added VIAddVersionKey: add version information resource to the installer
- Added AllowSkipFiles: set whether the user should be able to skip a file when overwriting failed
- Added LicenseForceSelection: checkbox or radio buttons on license page to let user agree with license or not
- Added SectionSetSize, SectionGetSize, SetCurInstType and GetCurInstType

- Replaced SetBkColor with SetCtlColors which can set text color too
- Added IfSilent and SetSilent to allow better /S interaction
- Added support for standard predefines: \${__DATE__}, \${__TIME__},
 \${__TIMESTAMP__}, \${__FILE__}, \${__LINE__}
- Added ifdiff and lastusd for SetOverwrite
- /o Switch for Section provides ability to unselect the section by default
- New parameter for DirText to set the browse dialog text
- Added RMDir /REBOOTOK: remove folders on reboot
- InstType /NOCUSTOM and /COMPONENTSONLYONCUSTOM work together
- Added base_dir for GetTempFileName

F.67.2.3 Minor Changes

- RO sections can now be in InstTypes too (defaults to old behavior)
- Increased limit of InstTypes to 32
- Improved macros & functions: UpgradeDLL, GetParent, GetParameters. Using the new versions is recommended.
- Fixed SetOutPath not setting current directory if the directory didn't exist before
- Components tree: fixed problems with sub-sections with RO sections as children, SF_EXPAND now refreshes the components tree and added SF_PSELECTED for partially selected sub-sections
- Improved AddBrandingImage: doesn't depend on the UI, can set image on the bottom and on the right, support for custom padding value
- Better installation directory verification
- Fixed all known problems with temporary files and directories
- Documentation fixes
- Minor bug fixes
- Code clean-ups and some more comments
- More...

F.67.2.4 Utilities and Plug-ins

- MakeNSISW 2.0: UI to define symbols, easy access to recent scripts, toolbar and more
- Zip2Exe 0.3: based on header files, improved interface, Modern UI support, new script code, improved folder detection
- InstallOptions 2.2: added LINK control, added EXTENDEDSELECT flag for list boxes which replaces MULTISELECT that now acts exactly as the real style flag ([double] click turns on or off selection), fixes
- BgImage plug-in stability fixes
- Added vPatch: patch generator and plug-in that applies the patches
- Banner plug-in improvements: responds to messages and some new /set tricks by brainsucker
- AdvSpalsh plug-in improvements: smaller, better transparency support and a possible bug fix
- System plug-in improvements: Unicode, GUID and COM support
- nsExec plug-in improvments: ability to run 16 bit code and tabs to spaces conversion
- Math plug-in
- Delphi unit for NSIS plug-ins

F.68 2.0 Beta 3

Released on March 16th, 2003

- Modern UI 1.63: Header bitmap support, new defines to change the description area, single macro for language selection dialog, more!
- New tutorial in the documentation
- define_if_last in Page command also works when a define has not been specified for all Page commands.
 This fixes the problem with the Modern UI "Click Next" / "Click
- Added SectionSetInstTypes and SectionGetInstTypes
- Reboot command does not force a reboot anymore (allows the user to save work)
- !if[n]def/!else fixes

Install" texts.

- LogSet on now really starts logging
- Cancel button available on all pages after the instfiles page but the last page unless /ENABLECANCEL was used in its Page command
- License page: No more limit on RTF size
- LangDLL: Option to auto-count number of languages, shell font support
- Page and UninstPage can not be used inside sections/functions
- CreateDirectory now uses the error flag
- EnumRegKey/Value output_var check fixed
- Updated translations
- Dreaded BSOD after plug-ins enumeration finally banished
- Minimize button and BGGradient and BGImage compatibility fixes
- WriteINIStr with empty value works again
- Added FlushINI

F.69 2.0 Beta 2

Released on February 26th, 2003

- NSIS Menu: links to all NSIS utilities, documentation and websites
- NSIS Update: check for new releases, download latest development files
- Dialer plugin to connect to the internet
- Improved filename validation
- VC7 compiler compatibility issues fixed
- CreateDirectory sets error flag
- InstallOptions: INI File State value fixed
- Updated translations
- Minor fixes

F.70 2.0 Beta 1

Released on February 9th, 2003

- User interface improvements: better order/grouping of controls, button selection problems fixed, added minimize box, transparent BrandingText
- LangString improvements: can be used before they have been defined, warning if not present in all language tables
- New version of the Modern User Interface with UI improvements, bugfixes and some small new features
- InstallOptions 2.0 with support for custom font and DPI settings, groupboxes, grouped controls, lots of bugfixes etc.
- No more random compression ratios
- SetOutPath now sets the working directory
- File names are validated, directory given by the user will now always work
- \$QUICKLAUNCH now works with SetShellVarContext all
- Automatically appended directory name in the directory selection dialog will no longer contain squares if not all characters are ASCII.
- Fixed a bug with remote drives and available space (\remote\drive)
- Plug-in function's names are now case insensitive
- Fixed a bug with specifying Icon twice
- Include dirs (!addincludedir)
- On leave function for pages
- Installers can now really contain more than one branding image
- All !if/!else/!endif problems should be solved now
- SetFont "MS Shell Dlg" adds the DS_SHELLFONT style
- Win9x rename on reboot now also works when the destination file does not exist
- DeleteRegKey works on default value
- nsExec can be called from an installer executed by CreateProcess
- WriteUninstaller sets error code
- Banner.dll compatible with Modern UI
- Fixed focus problems in MakeNSISw
- Updated and new translations

- Some new bitmaps for the MUI by Virtlink
- Faster and better MakeNSISw integration
- New .NET Framework detection function
- Minor bugfixes
- Code clean-ups (compiles on VC7)

F.71 2.0 Beta 0

Released on December 6th, 2002

- New paging system
- Added Page and UninstPage
- Removed .onNextPage, .onPrevPage, .onInitDialog
- New easier version of the Modern User Interface with better multilanguage support, InstallOptions integration, welcome and finish page etc.
- Custom pages no longer flicker when created
- Added accelerator keys
- Added LangString and LangStringUP for user defined multilingual strings
- Added support for transparent check marks
- Added InitPluginsDir
- Renamed nsisconf.nsi to nsisconf.nsh
- Added CreateShortcut comment/description parameter
- Splash.exe is now a plugin (splash.dll)
- Added new plugins: System, AdvSplash, nsExec, UserInfo, BgImage, Banner and StartMenu
- !ifdef and friends can now be used in macros
- SendMessage can send strings (put STR: before a param) and supports timeouts
- Right mouse button "Copy to clipboard" context menu for the Details window
- Plugin syntax now requires dll name, e.g. dll::func not just func
- Licence text receives initial focus (page up & down work immediately, return key still works)
- Made win9x move/delete on reboot support proper and function like on win2k
- Now always loads RichEdit v2 if present (links work on Win9x)
- DeleteRegKey now complains if given a third parameter (other than /ifempty)
- RegDLL now works with DLLs dependent on DLLs from the same directory

- \${LANG_langName} defined as the language id when loading a NLF
- Fixed a few bugs that caused the installer to load slower than before
- Uninstall can now get command line parameters too
- Added ReserveFile
- Added ExDLL.h
- Included makensisw 1.9 (lots of changes)
- Updated InstallOptions
- Updated ZIP2EXE
- Updated HTTP download plugin, NSIS-dl (timeout added, better proxy detection)
- A lot of new language files
- A lot of new icons and check marks
- Changed section flags values
- SectionSetFlags can now set bold too
- SectionSetFlags now works in silent installers too
- SectionSetName for sub sections doesn't need '-' in front of the name
- Added SetPluginUnload
- Hidden section can now use SectionIn too
- File /nonfatal switch added
- Plugins no longer add size to their containing section
- Defines can now be used inside define names (\${bla\${blo}})
- New docs format (WOOHA!)
- EXE header size a lot smaller, 33.5KB for bzip2 and 34KB for zlib
- Lots of other bugs fixed...

F.72 2.0 Alpha 7

Released on August 29th, 2002

- Can now select the language from .onInit (\$LANGUAGE is a variable)
- Added CreateFont
- Added .onMouseOverSection
- Added .onInitDialog and un.onInitDialog
- Added SetStaticBkColor
- Disabled UseOuterUlItem
- SendMessage WM_SETTEXT treats IParam as a string and not a number
- CopyFiles script message no longer always prints (silent)
- Custom install type text can now be changed
- ChangeUI can now change IDD_VERIFY
- Default license color now fits the user system
- Summary reports the right number of required sections
- Checkbox is only required in IDD DIR if logging is enabled
- Not using /LANG now really causes the script compiler to use the last used language
- Fixed a bug with MBCS and the uninstaller text
- MBCS to Unicode and Unicode to MBCS conversion bugs fixed
- Fixed a bug that caused RTF not to show on Windows 9x
- Added a dialog that shows up if compress whole is used and initial decompressing lasts longer than a second
- Updated to InstallOptions 1.3
- MakeNSISw 1.7 included
- Added modern UI by Joost Verburg (Examples\Modern UI)
- Added new full color icons by adni18
- Added Dutch, Korean, Russian, Swedish and Traditional Chinese language files
- Updated Spanish language file version (thanks to LsMoNKi)
- Added /TRIM(LEFT|RIGHT|CENTER) for BrandingText
- EXE header size down to 36.5KB
- Added yi-pixel.ico and yi-pixel-uninstall.ico by Jan T. Sott

- Fixed a bug with macros at the end of the file
- Sub-sections can now have defined names too
- Added LangDLL.dll plugin
- CallinstDLL /NOUNLOAD added (works on plugin calls too)

F.73 2.0 Alpha 6

Released on August 10th, 2002

- Fixed two bugs with the plug-in mechanism
- Fixed infinite loop bug

F.74 2.0 Alpha 5

Released on August 9th, 2002

- Added UseOuterUlltem
- Enhanced ChangeUI
- Enhanced SetDlgItemText
- Added one-section.nsi example file
- Optimized Ximon's code for plug-in DLLs (back to 37KB)
- If default user language doesn't fit exactly, will try to find primary language match
- Fixed some strings that got replaced (space available and required etc.)
- Compiles without MS Platform SDK
- Included Spanish.nlf
- Included MagicLime.exe by snowchyld

F.75 2.0 Alpha 4

Released on August 5th, 2002

- Added automatic detection, packing, unpacking and deletion of plugin dlls
- Added simplified calling syntax for plugin dlls
- Added PluginDir

F.76 2.0 Alpha 3

Released on August 4th, 2002

- Added LoadLanguageFile
- Added \$LANGUAGE
- Added /LANG option to string setters (Name, Caption, etc.)
- 'LogSet on' now builds a log file if not already created

F.77 2.0 Alpha 2

Released on July 31st, 2002

- Added \${NSISDIR}
- Added XPStyle
- Added SetFont
- Added ChangeUI
- Added AddBrandingImage, and SetBrandingImage
- Added SetCompressor (no more makensis-bz2.exe)
- Added LicenseBkColor
- 'SpaceTexts none' now causes no space texts to appear
- Icon and UninstallIcon now support icons of any type
- CheckBitmap now support bitmaps with any color table/depth
- Unused resources are removed before writing out the installer
- Documented !error and !warning
- Added !echo and !verbose
- Added force option for CRCCheck
- Updated to Makensisw 1.6
- Updated to InstallOptions 1.2
- Installer can now be compressed using UPX
- A hint is shown if a line is longer than the detail window
- Modern style folder select dialog
- License data can now be RTF
- Windowlcon is now handled in the compiler
- Removed debug version (it never worked anyway)
- Faster compile time (WIN32_LEAN_AND_MEAN)

F.78 2.0 Alpha 1

Released on July 9th, 2002

- Added expand node option to Section and SubSection
- Added preserve file attribute option to File command
- Copy-All bug fix in Makensisw
- Added NSISDIR to the define list
- Reorganized NSIS directory structure

F.79 2.0 Alpha 0

Released on May 22nd, 2002

- TreeView component list (care of Jeff Doozan)
- No more SectionDivider, but SubSection and SubSectionEnd.
- No more EnabledBitmap/DisabledBitmap, just CheckBitmap. (with tons of button states)
- Bugfixes of brokenness.
- Added ! for Section/SubSection to make bold.
- Made config.h have hacks for easier building for me. :)

F.80 Older Versions

NSIS 1.x version history

Appendix G: Building NSIS

NSIS version 2.07 introduced a new build system, based on SCons. The build system can build the entire NSIS package so you no longer need to build it project by project. It allows building using several simultaneous jobs, installation without an installer on both Windows and POSIX and easy compilation with debugging symbols.

The official release is built with MSVC6 Service Pack 5 (Mirror) with the Processor Pack and the February 2003 Platform SDK (5.2.3790.0).

- Building in General
- Building on Windows
- Building on POSIX
- Nightly Builds

G.1 Building in General

Source code is available in SVN and as a separate package with every NSIS distribution.

To build NSIS, Python and SCons must be installed. Currently, the supported version of SCons is version 1.2.0 and above. Any version of Python above 1.6 is supported.

NSIS uses the zlib compression library. As a consequence the header and library files of zlib must be installed.

In case these zlib development files aren't present then they could be installed via a package manager (apt-get, aptitude, rpm, yum) on POSIX platforms. Another option is to build zlib from scratch and install it.

For Windows it is recommended to download zlib from http://nsis.sf.net/Zlib. Extract the contents of this zip archive to a folder of your choice, e.g. C:\dev\zlib-1.2.7 and set an environment variable named zlib_w32 containing this path.

Alternatively the command line option ZLIB_W32 specifying the path can be passed to scons instead of the environment variable.

The header and library files are assumed to be in %ZLIB_W32%. In addition scons checks for zlib header files in %ZLIB_W32%\include, the import library zdll.lib in %ZLIB_W32%\lib and the dynamic link library zlib1.dll in %ZLIB_W32% respectively %ZLIB_W32%\lib.

To build, open a console, change the working directory to the root directory of NSIS and type scons. That's it. For example:

C:\>cd dev\nsis

```
C:\dev\nsis>scons
scons: Reading SConscript files ...
Using Microsoft tools configuration
Checking for main() in C library gdi32... (cached) yes
Checking for main() in C library user32... (cached) ye
Checking for main() in C library version... (cached) y
Checking for main() in C library pthread... (cached) n
Checking for main() in C library stdc++... (cached) no
Checking for main() in C library iconv... (cached) no
Checking for main() in C library libiconv... (cached)
scons: done reading SConscript files.
scons: Building targets ...
...
```

To install the built files, type:

```
scons PREFIX="C:\Program Files\NSIS" install
```

To create an installer (only on Windows), type:

```
scons dist-installer
```

To create a distribution zip file, type:

```
scons dist-zip
```

To create both, type:

```
scons dist
```

To get a complete list of options that the build system has to offer, type:

```
scons -h
```

To get a complete list of options SCons has to offer, type:

```
scons -H
```

G.2 Building on Windows

SCons will automatically detect Microsoft Visual C++. If you are looking for a free compiler to compile NSIS, we recommend Microsoft Visual C++ 2005 Express Edition.

When using the Microsoft Visual C++ Toolkit 2003, add MSTOOLKIT=yes to the build command line:

```
scons MST00LKIT=yes
```

In case of errors about the compiler or the Platform SDK not being found, use:

```
set MSSDK=C:\Path\To\Platform SDK
set VCToolkitInstallDir=C:\Path\To\VCToolkit
scons MST00LKIT=yes
```

The open-source MinGW can also be used to for building but this results in noticeably larger installers. Borland C++ or Open Watcom C/C++ might also work, but haven't been tested.

To compile the documentation as a CHM file, hhc.exe must be in the PATH. It is available as part of HTML Help Workshop.

To build NSIS Menu, install wxWidgets 2.8, create an environment variable named wxwin containing the path to the installation directory of wxWidgets, run contrib\NSIS Menu\wx\wxbuild.bat and build NSIS as usual.

Important notes for Microsoft Visual C++ 6.0 users: The 2003 Platform SDK must be installed before building, you can download it here or order it on CD. Because of flaws in the libraries distributed with Microsoft Visual C++ 6.0, not installing the Platform SDK will result in crashes when using the CopyFiles command. See this forum topic for more information. Installing the Processor Pack is highly recommended to decrease the size of the installer overhead.

Important note for Microsoft Visual C++ 2012 users: Installers generated using this release and beyond will not be able to run on Windows 9x or Windows 2000. The minimum requirement for 2012 is Windows XP.

G.3 Building on POSIX

As of NSIS 2.01, the compiler, makensis, also compiles on POSIX platforms. POSIX platforms include Linux, *BSD, Mac OS X and others. Since the generated installer will eventually run on Windows, a cross-compiler is needed in order to compile them.

The command line option XGCC_W32_PREFIX could be used to explicitly select a specific win32 targeted cross compiler. The value of XGCC_W32_PREFIX has to be the prefix of the toolchain. For example XGCC_W32_PREFIX=i686-w64-mingw32 would deploy the win32 targeted MinGW-w64 cross compiler if it is available on the build system.

If no cross-compiler is available, use the following:

scons SKIPSTUBS=all SKIPPLUGINS=all SKIPUTILS=all SKIP
 NSIS_CONFIG_CONST_DATA_PATH=no PREFIX=/path/to/e
 install-compiler

scons NSIS_CONFIG_CONST_DATA_PATH=no PREFIX=/path/to/e
 /path/to/extracted/zip/LibraryLocal

This should only build makensis and install it to the directory where a precompiled package, such as the nightly build or a zipped release version (nsis-x.xx.zip), is extracted. Note that the in order for this to work, the precompiled package must be compiled using the exact same sources as makensis. In particular, Source\exehead\config.h, the options passed to scons and Source\exehead\fileform.h must be identical. Nightly builds and zipped release versions are built with the default options.

To build a native version of NSIS Menu, install wxWidgets 2.8 and build as usual. wx-config must be in the path.

G.4 Nightly Builds

There is no need to manually build the latest SVN version for Windows. A nightly build is available. The nightly build is automatically generated every night, using the latest version of the source code from SVN. There is no official nightly build for other platforms.

Appendix H: Credits

- Programmers
- Testers
- Designers
- Translators
- Writers

H.1 Programmers

Justin Frankel aka 0xDEADBEEF

• Creating the all mighty NSIS

Amir "make me stop" Szekely aka KiCHiK

- Multilingual NSIS
- RTF license text
- The new paging system
- Full color support for icons and bitmaps
- Branding image
- Customizable UI
- One makensis.exe for both zlib and bzip2

Joost Verburg

- Modern User Interface
- NSIS website
- NSIS Menu
- System for DLL/TLB library setup
- NSIS Update for NSIS distribution (original version by Nathan Purciful)

Robert Rainwater

- MakeNSISW
- New documentation format
- Enhancing the TreeView
- Reorganizing NSIS directory structure

Dave "bit-by-bit" Laundon aka eccles

Massive optimizing

Ximon Eighteen aka Sunjammer

- The new plug-ins system
- "Copy to clipboard" context menu for the Details window
- License text initial focus

Ramon aka Ramon18

- Version information resource commands
- Named user variables
- Lots of UI fixes
- InstallOptions improvements

Jim Park

• Unicode support

Olivier Marcoux aka Wizou

Unicode merge

nnop@newmail.ru

Ryan Geiss

Andras Varga

Drew Davidson

Peter Windridge

Yaroslav Faybishenko

Jeff Doozan

NSIS 2's new TreeView

Nike (nike@sendmail.ru)

• HTML Help support for Halibut

Diego Pedroso aka deguix

• New NSIS Wiki

Shengalts Aleksander aka Instructor

Stuart Welch aka Afrow UK

David Weiss aka Comm@nder21

Anders Kjersem

• NSIS 3 patron saint

H.2 Testers

Jason Ross aka JasonFriday13

• NSIS 3 POSIX support

H.3 Designers

Nikos Adamamas

• The new modern icons

Jan T. Sott / whyEye.org

• Lots of icons and check marks

H.4 Translators

Albanian - Besnik Bleta

Afrikaans - Friedel Wolff

Arabic - asdfuae, Rami Kattan

Armenian - Hrant Ohanyan

Asturian - Marcos (marcoscostales@gmail.com)

Basque - Iñaki San Vicente

Belarusian - Sitnikov Vjacheslav

Bosnian - Salih CAVKIC

Breton - Korvigelloù An Drouizig

Bulgarian - Asparouh Kalyandjiev, Plamen Penkov

Catalan - falanko

Croatian - Igor Ostriz, Vedran "RIV@NVX" Miletic

Czech - T.V. Zuggy, SELiCE

Danish - Christopher, Casper Bergenstoff, Claus Futtrup

Dutch - Hendri Adriaens, Joost Verburg

Esperanto - Felipe Castro

Estonian - izzo

Farsi - Masoud Alinaqian, FzerorubigD, Elnaz Sarbar

Finnish - AKX, Eclipser

French - veekee, Sebastien Delahaye, Jerome Charaoui

Galician - Ramon Flores

Georgian - David Huriev

German - L.King, K. Windszus, R. Bisswanger, M. Simmack, Tim Kosse

Greek - Makidis N. Michael

Hebrew - Amir Szekely (aka KiCHiK), Yaron Shahrabani

Hungarian - Soft-Trans Bt., Jozsef Tamas Herczeg, Lajos Molnar (Orfanik)

Icelandic - Gretar Orri Kristinsson

Indonesian - ariel825010106

Italian - Orfanik, sanface, Alessandro Staltari, Lorenzo Bevilacqua

Japanese - Dnanako, Takahiro Yoshimura

Khmer - yi sophally

Korean - dTomoyo, linak, koder

Kurdish - Erdal Ronahi

Latvian - Valdis Griíis, Kristaps Meòìelis

Lithuanian - NorCis, Vytautas Krivickas, Danielius Scepanskis

Luxembourgish - Jo Hoeser

Macedonian - Sasko Zdravkin

Mongolian - Bayarsaikhan Enkhtaivan

Norwegian - Jonas Christoffer Lindstrom, Jan Ivar Beddari

Norwegian Nynorsk - Vebjørn Sture

Pashto - Pakhtosoft

Polish - Piotr Murawski, Rafał Lampe, cube, SYSTEMsoft Group, Marek Stępień, Mateusz Gola, Paweł Porwisz

Portuguese - DragonSoull, Dre', Ramon

Portuguese Brasil - Layout do Brasil, deguix

Romanian - Sorin Sbarnea, Cristian Pirvu, George Radu, Vlad Rusu

Russian - Sergey `Timon` Kusnetsov, Nik Medved, Scam, THRaSH, Dmitry Yerokhin

Serbian - Srdjan Obucina

Serbian Latin - Srdjan Obucina, Vladan Obradovic

Slovak - trace, Kypec, Marián Hikaník

Slovenian - Janez Dolinar, Martin Sebotnjak

Spanish - MoNKi, Lobo Lunar, Darwin Rodrigo Toledo Cáceres

Swedish - Peter Gustafsson, Magnus Bonnevier, Rickard Angbratt

Thai - SoKoOLz, TuW@nNu (asdfuae)

Traditional & Simplified Chinese - Kii Ali

Turkish - Bertan Kodamanoglu, Cagatay Dilsiz, Fatih BOY

Ukrainian - Yuri Holubow, Nash-Soft

Uzbek - Emil Garipov (emil.garipov@gmail.com)

Valencian - Bernardo Arlandis Mañó

Vietnamese - Clytie Siddall

Welsh - Rhoslyn Prys, Meddal.com

H.5 Writers

Sebastian Armbrust aka flizebogen

• Tutorial

Appendix I: License

- Copyright
- Applicable licenses
- zlib/libpng license
- bzip2 license
- Common Public License version 1.0
- Special exception for LZMA compression module

I.1 Copyright

Copyright (C) 1999-2016 Contributors

More detailed copyright information can be found in the individual source code files.

I.2 Applicable licenses

- All NSIS source code, plug-ins, documentation, examples, header files and graphics, with the exception of the compression modules and where otherwise noted, are licensed under the zlib/libpng license.
- The zlib compression module for NSIS is licensed under the zlib/libpng license.
- The bzip2 compression module for NSIS is licensed under the bzip2 license.
- The Izma compression module for NSIS is licensed under the Common Public License version 1.0.

I.3 zlib/libpng license

This software is provided 'as-is', without any express or implied warranty. In no event will the authors be held liable for any damages arising from the use of this software.

Permission is granted to anyone to use this software for any purpose, including commercial applications, and to alter it and redistribute it freely, subject to the following restrictions:

- 1. The origin of this software must not be misrepresented; you must not claim that you wrote the original software. If you use this software in a product, an acknowledgment in the product documentation would be appreciated but is not required.
- 2. Altered source versions must be plainly marked as such, and must not be misrepresented as being the original software.
- 3. This notice may not be removed or altered from any source distribution.

I.4 bzip2 license

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

- 1. Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.
- 2. The origin of this software must not be misrepresented; you must not claim that you wrote the original software. If you use this software in a product, an acknowledgment in the product documentation would be appreciated but is not required.
- 3. Altered source versions must be plainly marked as such, and must not be misrepresented as being the original software.
- 4. The name of the author may not be used to endorse or promote products derived from this software without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE AUTHOR "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE AUTHOR BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

Julian Seward, Cambridge, UK.

jseward@acm.org

I.5 Common Public License version 1.0

THE ACCOMPANYING PROGRAM IS PROVIDED UNDER THE TERMS OF THIS COMMON PUBLIC LICENSE ("AGREEMENT"). ANY USE, REPRODUCTION OR DISTRIBUTION OF THE PROGRAM CONSTITUTES RECIPIENT'S ACCEPTANCE OF THIS AGREEMENT.

1. DEFINITIONS

"Contribution" means:

- a) in the case of the initial Contributor, the initial code and documentation distributed under this Agreement, and b) in the case of each subsequent Contributor:
- i) changes to the Program, and
- ii) additions to the Program;

where such changes and/or additions to the Program originate from and are distributed by that particular Contributor. A Contribution 'originates' from a Contributor if it was added to the Program by such Contributor itself or anyone acting on such Contributor's behalf. Contributions do not include additions to the Program which: (i) are separate modules of software distributed in conjunction with the Program under their own license agreement, and (ii) are not derivative works of the Program.

"Contributor" means any person or entity that distributes the Program.

"Licensed Patents" mean patent claims licensable by a Contributor which are necessarily infringed by the use or sale of its Contribution alone or when combined with the Program.

"Program" means the Contributions distributed in accordance with this Agreement.

"Recipient" means anyone who receives the Program under this

Agreement, including all Contributors.

2. GRANT OF RIGHTS

- a) Subject to the terms of this Agreement, each Contributor hereby grants Recipient a non-exclusive, worldwide, royalty-free copyright license to reproduce, prepare derivative works of, publicly display, publicly perform, distribute and sublicense the Contribution of such Contributor, if any, and such derivative works, in source code and object code form.
- b) Subject to the terms of this Agreement, each Contributor hereby grants Recipient a non-exclusive, worldwide, royalty-free patent license under Licensed Patents to make, use, sell, offer to sell, import and otherwise transfer the Contribution of such Contributor, if any, in source code and object code form. This patent license shall apply to the combination of the Contribution and the Program if, at the time the Contribution is added by the Contributor, such addition of the Contribution causes such combination to be covered by the Licensed Patents. The patent license shall not apply to any other combinations which include the Contribution. No hardware per se is licensed hereunder.
- c) Recipient understands that although each Contributor grants the licenses to its Contributions set forth herein, no assurances are provided by any Contributor that the Program does not infringe the patent or other intellectual property rights of any other entity. Each Contributor disclaims any liability to Recipient for claims brought by any other entity based on infringement of intellectual property rights or otherwise. As a condition to exercising the rights and licenses granted hereunder, each Recipient hereby assumes sole responsibility to secure any other intellectual property rights needed, if any. For example, if a third party patent license is required to allow Recipient to distribute the Program, it is Recipient's responsibility to acquire that license before distributing the Program.
- d) Each Contributor represents that to its knowledge it has sufficient copyright rights in its Contribution, if any, to grant the copyright license set forth in this Agreement.

3. REQUIREMENTS

A Contributor may choose to distribute the Program in object code form under its own license agreement, provided that:

- a) it complies with the terms and conditions of this Agreement; and
- b) its license agreement:
- i) effectively disclaims on behalf of all Contributors all warranties and conditions, express and implied, including warranties or conditions of title and non-infringement, and implied warranties or conditions of merchantability and fitness for a particular purpose;
- ii) effectively excludes on behalf of all Contributors all liability for damages, including direct, indirect, special, incidental and consequential damages, such as lost profits;
- iii) states that any provisions which differ from this Agreement are offered by that Contributor alone and not by any other party; and
- iv) states that source code for the Program is available from such Contributor, and informs licensees how to obtain it in a reasonable manner on or through a medium customarily used for software exchange.

When the Program is made available in source code form:

- a) it must be made available under this Agreement; and
- b) a copy of this Agreement must be included with each copy of the Program.

Contributors may not remove or alter any copyright notices contained within the Program.

Each Contributor must identify itself as the originator of its Contribution, if any, in a manner that reasonably allows subsequent Recipients to identify the originator of the Contribution.

4. COMMERCIAL DISTRIBUTION

Commercial distributors of software may accept certain responsibilities with respect to end users, business partners and the like. While this license is intended to facilitate the commercial use of the Program, the Contributor who includes the Program in a commercial product offering should do so in a manner which does not create potential liability for other Contributors. Therefore, if a Contributor includes the Program in a commercial product offering, such Contributor ("Commercial Contributor") hereby agrees to defend and indemnify every other Contributor ("Indemnified Contributor") against any losses, damages and costs (collectively "Losses") arising from claims, lawsuits and other legal actions brought by a third party against the Indemnified Contributor to the extent caused by the acts or omissions of such Commercial Contributor in connection with its distribution of the Program in a commercial product offering. The obligations in this section do not apply to any claims or Losses relating to any actual or alleged intellectual property infringement. In order to qualify, an Indemnified Contributor must: a) promptly notify the Commercial Contributor in writing of such claim, and b) allow the Commercial Contributor to control, and cooperate with the Commercial Contributor in, the defense and any related settlement negotiations. The Indemnified Contributor may participate in any such claim at its own expense.

For example, a Contributor might include the Program in a commercial product offering, Product X. That Contributor is then a Commercial Contributor. If that Commercial Contributor then makes performance claims, or offers warranties related to Product X, those performance claims and warranties are such Commercial Contributor's responsibility alone. Under this section, the Commercial Contributor would have to defend claims against the other Contributors related to those performance claims and warranties, and if a court requires any other Contributor to pay any damages as a result, the Commercial Contributor must pay those damages.

5. NO WARRANTY

EXCEPT AS EXPRESSLY SET FORTH IN THIS AGREEMENT, THE PROGRAM IS PROVIDED ON AN "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, EITHER EXPRESS

OR IMPLIED INCLUDING, WITHOUT LIMITATION, ANY WARRANTIES OR CONDITIONS OF TITLE, NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Each Recipient is solely responsible for determining the appropriateness of using and distributing the Program and assumes all risks associated with its exercise of rights under this Agreement, including but not limited to the risks and costs of program errors, compliance with applicable laws, damage to or loss of data, programs or equipment, and unavailability or interruption of operations.

6. DISCLAIMER OF LIABILITY

EXCEPT AS EXPRESSLY SET FORTH IN THIS AGREEMENT, NEITHER RECIPIENT NOR ANY CONTRIBUTORS SHALL HAVE ANY LIABILITY FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING WITHOUT LIMITATION LOST PROFITS), HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OR DISTRIBUTION OF THE PROGRAM OR THE EXERCISE OF ANY RIGHTS GRANTED HEREUNDER, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

7. GENERAL

If any provision of this Agreement is invalid or unenforceable under applicable law, it shall not affect the validity or enforceability of the remainder of the terms of this Agreement, and without further action by the parties hereto, such provision shall be reformed to the minimum extent necessary to make such provision valid and enforceable.

If Recipient institutes patent litigation against a Contributor with respect to a patent applicable to software (including a cross-claim or counterclaim in a lawsuit), then any patent licenses granted by that Contributor to such Recipient under this Agreement shall terminate as of the date such litigation is filed. In addition, if Recipient institutes patent litigation against any entity (including a cross-claim or counterclaim in a lawsuit) alleging

that the Program itself (excluding combinations of the Program with other software or hardware) infringes such Recipient's patent(s), then such Recipient's rights granted under Section 2(b) shall terminate as of the date such litigation is filed.

All Recipient's rights under this Agreement shall terminate if it fails to comply with any of the material terms or conditions of this Agreement and does not cure such failure in a reasonable period of time after becoming aware of such noncompliance. If all Recipient's rights under this Agreement terminate, Recipient agrees to cease use and distribution of the Program as soon as reasonably practicable. However, Recipient's obligations under this Agreement and any licenses granted by Recipient relating to the Program shall continue and survive.

Everyone is permitted to copy and distribute copies of this Agreement, but in order to avoid inconsistency the Agreement is copyrighted and may only be modified in the following manner. The Agreement Steward reserves the right to publish new versions (including revisions) of this Agreement from time to time. No one other than the Agreement Steward has the right to modify this Agreement. IBM is the initial Agreement Steward. IBM may assign the responsibility to serve as the Agreement Steward to a suitable separate entity. Each new version of the Agreement will be given a distinguishing version number. The Program (including Contributions) may always be distributed subject to the version of the Agreement under which it was received. In addition, after a new version of the Agreement is published, Contributor may elect to distribute the Program (including its Contributions) under the new version. Except as expressly stated in Sections 2(a) and 2(b) above, Recipient receives no rights or licenses to the intellectual property of any Contributor under this Agreement, whether expressly, by implication, estoppel or otherwise. All rights in the Program not expressly granted under this Agreement are reserved.

This Agreement is governed by the laws of the State of New York and the intellectual property laws of the United States of America. No party to this Agreement will bring a legal action under this Agreement more than one year after the cause of action arose. Each party waives its rights to a jury trial in any resulting litigation.

I.6 Special exception for LZMA compression module

Igor Pavlov and Amir Szekely, the authors of the LZMA compression module for NSIS, expressly permit you to statically or dynamically link your code (or bind by name) to the files from the LZMA compression module for NSIS without subjecting your linked code to the terms of the Common Public license version 1.0. Any modifications or additions to files from the LZMA compression module for NSIS, however, are subject to the terms of the Common Public License version 1.0.